LilyPond

Il compositore tipografico per la musica

Guida alla Notazione

Il team di sviluppo di LilyPond

Questo manuale costituisce la guida di riferimento per tutti gli aspetti relativi alla notazione musicale in LilyPond versione 2.18.2. Si presuppone che il lettore conosca il materiale esposto nel Sezione "Manuale di Apprendimento" in *Manuale di Apprendimento*.

Per maggiori informazioni su come questo manuale si integra col resto della documentazione, o per leggere questo manuale in altri formati, si veda Sezione "Manuali" in *Informazioni generali*.

Se ti manca qualche manuale, puoi trovare la completa documentazione all'indirizzo `http://www.lilypond.org/`.

Copyright © 1999–2012 degli autori.

La traduzione della seguente nota di copyright è gentilmente offerta per le persone che non parlano inglese, ma solo la nota in inglese ha valore legale.

The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.

> E' garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della Licenza per Documentazione Libera GNU, Versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation; senza alcuna sezione non modificabile. Una copia della licenza è acclusa nella sezione intitolata "Licenza per Documentazione Libera GNU".
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections. A copy of the license is included in the section entitled "GNU Free Documentation License".

Per la versione di LilyPond 2.18.2

Sommario

1 Notazione musicale ... 1

1.1 Altezze ... 1
1.1.1 Inserimento delle altezze ... 1
Ottava assoluta ... 1
Ottava relativa ... 2
Alterazioni ... 5
Nomi delle note in altre lingue ... 7
1.1.2 Modifica di più altezze ... 9
Controlli di ottava ... 9
Trasposizione ... 10
Inversione ... 13
Retrogradazione ... 14
Trasposizioni modali ... 14
1.1.3 Aspetto delle altezze ... 16
Chiave ... 17
Armatura di chiave ... 20
Segni di ottavazione ... 23
Trasporto strumentale ... 24
Alterazioni automatiche ... 26
Ambitus ... 33
1.1.4 Teste di nota ... 35
Teste di nota speciali ... 35
Testa di nota con nome della nota ... 37
Teste di nota a forma variabile ... 38
Improvvisazione ... 41
1.2 Ritmi ... 42
1.2.1 Inserimento delle durate ... 43
Durata ... 43
Gruppi irregolari ... 45
Scalare le durate ... 49
Legature di valore ... 50
1.2.2 Inserimento delle pause ... 54
Pause ... 54
Pause invisibili ... 56
Pause d'intero ... 57
1.2.3 Aspetto dei ritmi ... 61
Indicazione di tempo ... 61
Indicazioni metronomiche ... 66
Anacrusi ... 69
Musica in tempo libero ... 70
Notazione polimetrica ... 72
Divisione automatica delle note ... 75
Mostrare i ritmi della melodia ... 76
1.2.4 Travature ... 78
Travature automatiche ... 78
Impostare il comportamento delle travature automatiche ... 81
Travature manuali ... 89
Travature a raggiera ... 92

1.2.5 Battute ... 93
Stanghette ... 93
Numeri di battuta ... 100
Controlli di battuta e del numero di battuta ... 104
Segni di chiamata ... 105
1.2.6 Questioni ritmiche particolari ... 107
Abbellimenti ... 107
Allineamento sulle cadenze ... 113
Gestione del tempo ... 114
1.3 Segni di espressione ... 115
1.3.1 Segni di espressione collegati alle note ... 115
Articolazioni e abbellimenti ... 115
Dinamiche ... 118
Nuove indicazioni dinamiche ... 124
1.3.2 Indicazioni espressive curvilinee ... 126
Legature di portamento ... 126
Legature di frase ... 129
Respiri ... 130
Portamenti indeterminati discendenti (cadute) e ascendenti ... 132
1.3.3 Indicazioni espressive lineari ... 132
Glissando ... 133
Arpeggio ... 137
Trilli ... 140
1.4 Ripetizioni ... 142
1.4.1 Ripetizioni lunghe ... 143
Ripetizioni normali ... 143
Indicazioni di ripetizione manuali ... 151
Ripetizioni ricopiate ... 153
1.4.2 Ripetizioni brevi ... 155
Ripetizioni con percentuale ... 155
Ripetizioni con tremolo ... 157
1.5 Note simultanee ... 159
1.5.1 Una voce ... 159
Note in un accordo ... 159
Ripetizione di un accordo ... 161
Espressioni simultanee ... 163
Cluster ... 164
1.5.2 Più voci ... 164
Polifonia su un solo rigo ... 164
Stili di voce ... 168
Risoluzione delle collisioni ... 168
Combinazione automatica delle parti ... 173
Scrivere la musica in parallelo ... 178
1.6 Notazione del rigo ... 180
1.6.1 Aspetto del rigo ... 181
Istanziare nuovi righi ... 181
Raggruppare i righi ... 182
Gruppi di righi annidati ... 186
Separare i sistemi ... 187
1.6.2 Modificare singoli righi ... 188
Simbolo del rigo ... 188
Righi ossia ... 192
Nascondere i righi ... 195
1.6.3 Scrittura delle parti ... 198

Nomi degli strumenti ... 199
Citare altre voci ... 202
Formattazione delle notine ... 205
1.7 Note editoriali ... 210
1.7.1 Interne al rigo ... 210
Scelta della dimensione del tipo di carattere ... 210
Indicazioni di diteggiatura ... 212
Note nascoste ... 214
Colorare gli oggetti ... 215
Parentesi ... 216
Gambi ... 217
1.7.2 Esterne al rigo ... 218
Nuvoletta di aiuto ... 218
Linee della griglia ... 219
Parentesi analitiche ... 221
1.8 Testo ... 222
1.8.1 Inserimento del testo ... 223
Scritte ... 223
Estensori del testo ... 224
Indicazioni testuali ... 226
Testo separato ... 228
1.8.2 Formattazione del testo ... 230
Introduzione al testo a margine ... 230
Scelta del tipo di carattere e della dimensione ... 231
Allineamento del testo ... 234
Notazione grafica nel blocco markup ... 237
Notazione musicale nel blocco markup ... 240
Testo formattato su più pagine ... 242
1.8.3 Tipi di carattere ... 243
Tipi di carattere in dettaglio ... 243
Tipi di carattere per singolo oggetto ... 245
Tipi di carattere per l'intero documento ... 245

2 Specialist notation ... 247

2.1 Vocal music ... 247
2.1.1 Common notation for vocal music ... 247
References for vocal music ... 247
Entering lyrics ... 248
Aligning lyrics to a melody ... 249
Automatic syllable durations ... 251
Manual syllable durations ... 253
Multiple syllables to one note ... 255
Multiple notes to one syllable ... 255
Extenders and hyphens ... 259
2.1.2 Techniques specific to lyrics ... 259
Working with lyrics and variables ... 259
Placing lyrics vertically ... 261
Placing syllables horizontally ... 265
Lyrics and repeats ... 267
Divisi lyrics ... 275
Polyphony with shared lyrics ... 276
2.1.3 Stanzas ... 278
Adding stanza numbers ... 278
Adding dynamics marks to stanzas ... 278

Adding singers' names to stanzas 279
Stanzas with different rhythms 279
Printing stanzas at the end 282
Printing stanzas at the end in multiple columns 283
2.1.4 Songs 285
References for songs 285
Lead sheets 285
2.1.5 Choral 286
References for choral 286
Score layouts for choral 287
Divided voices 288
2.1.6 Opera and stage musicals 289
References for opera and stage musicals 289
Character names 290
Musical cues 292
Spoken music 296
Dialogue over music 296
2.1.7 Chants psalms and hymns 297
References for chants and psalms 297
Setting a chant 297
Pointing a psalm 304
Partial measures in hymn tunes 307
2.1.8 Ancient vocal music 309
2.2 Keyboard and other multi-staff instruments 310
2.2.1 Common notation for keyboards 311
References for keyboards 311
Changing staff manually 312
Changing staff automatically 313
Staff-change lines 315
Cross-staff stems 315
2.2.2 Piano 317
Piano pedals 317
2.2.3 Accordion 318
Discant symbols 318
2.2.4 Harp 319
References for harps 319
Harp pedals 319
2.3 Unfretted string instruments 320
2.3.1 Common notation for unfretted strings 321
References for unfretted strings 321
Bowing indications 321
Harmonics 322
Snap (Bartók) pizzicato 323
2.4 Fretted string instruments 323
2.4.1 Common notation for fretted strings 324
References for fretted strings 324
String number indications 324
Default tablatures 326
Custom tablatures 339
Fret diagram markups 342
Predefined fret diagrams 351
Automatic fret diagrams 361
Right-hand fingerings 364
2.4.2 Guitar 366

Indicating position and barring ... 366
Indicating harmonics and dampened notes ... 366
Indicating power chords ... 368
2.4.3 Banjo ... 369
Banjo tablatures ... 369
2.5 Percussion ... 370
2.5.1 Common notation for percussion ... 370
References for percussion ... 370
Basic percussion notation ... 370
Drum rolls ... 371
Pitched percussion ... 372
Percussion staves ... 372
Custom percussion staves ... 374
Ghost notes ... 378
2.6 Wind instruments ... 378
2.6.1 Common notation for wind instruments ... 379
References for wind instruments ... 379
Fingerings ... 380
2.6.2 Bagpipes ... 382
Bagpipe definitions ... 382
Bagpipe example ... 382
2.6.3 Woodwinds ... 384
2.6.3.1 Woodwind diagrams ... 384
2.7 Chord notation ... 392
2.7.1 Chord mode ... 392
Chord mode overview ... 392
Common chords ... 393
Extended and altered chords ... 395
2.7.2 Displaying chords ... 397
Printing chord names ... 398
Customizing chord names ... 400
2.7.3 Figured bass ... 405
Introduction to figured bass ... 406
Entering figured bass ... 407
Displaying figured bass ... 409
2.8 Contemporary music ... 412
2.8.1 Pitch and harmony in contemporary music ... 412
References for pitch and harmony in contemporary music ... 412
Microtonal notation ... 412
Contemporary key signatures and harmony ... 412
2.8.2 Contemporary approaches to rhythm ... 412
References for contemporary approaches to rhythm ... 412
Tuplets in contemporary music ... 412
Contemporary time signatures ... 412
Extended polymetric notation ... 412
Beams in contemporary music ... 412
Bar lines in contemporary music ... 412
2.8.3 Graphical notation ... 412
2.8.4 Contemporary scoring techniques ... 412
2.8.5 New instrumental techniques ... 413
2.8.6 Further reading and scores of interest ... 413
Books and articles on contemporary musical notation ... 413
Scores and musical examples ... 413
2.9 Ancient notation ... 413

2.9.1 Overview of the supported styles . . . 414
2.9.2 Ancient notation—common features . . . 415
Pre-defined contexts . . . 415
Ligatures . . . 415
Custodes . . . 416
2.9.3 Typesetting mensural music . . . 417
Mensural contexts . . . 417
Mensural clefs . . . 417
Mensural time signatures . . . 419
Mensural note heads . . . 420
Mensural flags . . . 420
Mensural rests . . . 421
Mensural accidentals and key signatures . . . 422
Annotational accidentals (*musica ficta*) . . . 422
White mensural ligatures . . . 423
2.9.4 Typesetting Gregorian chant . . . 424
Gregorian chant contexts . . . 424
Gregorian clefs . . . 425
Gregorian accidentals and key signatures . . . 426
Divisiones . . . 426
Gregorian articulation signs . . . 427
Augmentum dots (*morae*) . . . 428
Gregorian square neume ligatures . . . 428
2.9.5 Typesetting Kievan square notation . . . 435
Kievan contexts . . . 435
Kievan clefs . . . 436
Kievan notes . . . 436
Kievan accidentals . . . 437
Kievan bar line . . . 437
Kievan melismata . . . 438
2.9.6 Working with ancient music—scenarios and solutions . . . 439
Incipits . . . 439
Mensurstriche layout . . . 439
Transcribing Gregorian chant . . . 439
Ancient and modern from one source . . . 442
Editorial markings . . . 442
2.10 World music . . . 443
2.10.1 Common notation for non-Western music . . . 443
Extending notation and tuning systems . . . 443
2.10.2 Arabic music . . . 444
References for Arabic music . . . 444
Arabic note names . . . 444
Arabic key signatures . . . 445
Arabic time signatures . . . 447
Arabic music example . . . 447
Further reading for Arabic music . . . 448
2.10.3 Turkish classical music . . . 449
References for Turkish classical music . . . 449
Turkish note names . . . 449

3 General input and output 450

3.1 Input structure 450
3.1.1 Structure of a score 450
3.1.2 Multiple scores in a book 451
3.1.3 Multiple output files from one input file 452
3.1.4 Output file names 453
3.1.5 File structure 454
3.2 Titles and headers 456
3.2.1 Creating titles headers and footers 456
Titles explained 456
Default layout of bookpart and score titles 459
Default layout of headers and footers 462
3.2.2 Custom titles headers and footers 463
Custom text formatting for titles 463
Custom layout for titles 464
Custom layout for headers and footers 467
3.2.3 Creating footnotes 468
Footnotes in music expressions 468
Footnotes in stand-alone text 474
3.2.4 Reference to page numbers 477
3.2.5 Table of contents 478
3.3 Working with input files 480
3.3.1 Including LilyPond files 480
3.3.2 Different editions from one source 481
Using variables 482
Using tags 483
Using global settings 486
3.3.3 Special characters 486
Text encoding 486
Unicode 487
ASCII aliases 488
3.4 Controlling output 489
3.4.1 Extracting fragments of music 489
3.4.2 Skipping corrected music 489
3.4.3 Alternative output formats 490
3.4.4 Replacing the notation font 490
3.5 MIDI output 491
3.5.1 Creating MIDI files 491
3.5.2 MIDI Instruments 494
3.5.3 What goes into the MIDI output? 494
Supported in MIDI 494
Unsupported in MIDI 494
3.5.4 Repeats in MIDI 495
3.5.5 Controlling MIDI dynamics 495
Dynamic marks 496
Overall MIDI volume 496
Equalizing different instruments (i) 497
Equalizing different instruments (ii) 498
3.5.6 Percussion in MIDI 499
3.5.7 The Articulate script 500
3.6 Extracting musical information 500
3.6.1 Displaying LilyPond notation 500
3.6.2 Displaying scheme music expressions 501
3.6.3 Saving music events to a file 501

4 Spacing issues 502

4.1 Page layout 502
4.1.1 The \paper block 502
4.1.2 Paper size and automatic scaling 503
Setting the paper size 503
Automatic scaling to paper size 504
4.1.3 Fixed vertical spacing \paper variables 504
4.1.4 Flexible vertical spacing \paper variables 505
Structure of flexible vertical spacing alists 505
List of flexible vertical spacing \paper variables 506
4.1.5 Horizontal spacing \paper variables 507
\paper variables for widths and margins 507
\paper variables for two-sided mode 508
\paper variables for shifts and indents 509
4.1.6 Other \paper variables 509
\paper variables for line breaking 509
\paper variables for page breaking 510
\paper variables for page numbering 511
Miscellaneous \paper variables 511
4.2 Score layout 512
4.2.1 The \layout block 512
4.2.2 Setting the staff size 514
4.3 Breaks 515
4.3.1 Line breaking 515
4.3.2 Page breaking 517
4.3.3 Optimal page breaking 518
4.3.4 Optimal page turning 518
4.3.5 Minimal page breaking 519
4.3.6 One-line page breaking 519
4.3.7 Explicit breaks 519
4.3.8 Using an extra voice for breaks 521
4.4 Vertical spacing 523
4.4.1 Flexible vertical spacing within systems 523
Within-system spacing properties 523
Spacing of ungrouped staves 526
Spacing of grouped staves 527
Spacing of non-staff lines 529
4.4.2 Explicit staff and system positioning 530
4.4.3 Vertical collision avoidance 537
4.5 Horizontal spacing 538
4.5.1 Horizontal spacing overview 538
4.5.2 New spacing area 540
4.5.3 Changing horizontal spacing 540
4.5.4 Line length 542
4.5.5 Proportional notation 543
4.6 Fitting music onto fewer pages 549
4.6.1 Displaying spacing 549
4.6.2 Changing spacing 550

5 Changing defaults ... 553

5.1 Interpretation contexts ... 553
5.1.1 Contexts explained ... 553
Output definitions - blueprints for contexts ... 553
Score - the master of all contexts ... 554
Top-level contexts - staff containers ... 554
Intermediate-level contexts - staves ... 554
Bottom-level contexts - voices ... 555
5.1.2 Creating and referencing contexts ... 555
5.1.3 Keeping contexts alive ... 558
5.1.4 Modifying context plug-ins ... 561
5.1.5 Changing context default settings ... 563
Changing all contexts of the same type ... 563
Changing just one specific context ... 566
Order of precedence ... 567
5.1.6 Defining new contexts ... 568
5.1.7 Context layout order ... 570
5.2 Explaining the Internals Reference ... 572
5.2.1 Navigating the program reference ... 572
5.2.2 Layout interfaces ... 572
5.2.3 Determining the grob property ... 574
5.2.4 Naming conventions ... 574
5.3 Modifying properties ... 575
5.3.1 Overview of modifying properties ... 575
5.3.2 The `\set` command ... 575
5.3.3 The `\override` command ... 577
5.3.4 The `\tweak` command ... 579
5.3.5 `\set` vs. `\override` ... 581
5.3.6 Modifying alists ... 581
5.4 Useful concepts and properties ... 583
5.4.1 Input modes ... 583
5.4.2 Direction and placement ... 585
Articulation direction indicators ... 585
The direction property ... 585
5.4.3 Distances and measurements ... 586
5.4.4 Staff symbol properties ... 586
5.4.5 Spanners ... 587
Using the `spanner-interface` ... 587
Using the `line-spanner-interface` ... 589
5.4.6 Visibility of objects ... 592
Removing the stencil ... 592
Making objects transparent ... 592
Painting objects white ... 593
Using break-visibility ... 593
Special considerations ... 595
5.4.7 Line styles ... 598
5.4.8 Rotating objects ... 598
Rotating layout objects ... 598
Rotating markup ... 599
5.5 Advanced tweaks ... 599
5.5.1 Aligning objects ... 600
Setting `X-offset` and `Y-offset` directly ... 600
Using the `side-position-interface` ... 601
Using the `self-alignment-interface` ... 601

Using the `break-alignable-interface` ... 602
5.5.2 Vertical grouping of grobs ... 604
5.5.3 Modifying stencils ... 604
5.5.4 Modifying shapes ... 605
Modifying ties and slurs ... 605
5.5.5 Modifying broken spanners ... 609
Using `\alterBroken` ... 609
5.5.6 Unpure-pure containers ... 611
5.6 Using music functions ... 612
5.6.1 Substitution function syntax ... 612
5.6.2 Substitution function examples ... 613

Appendice A Notation manual tables ... 615

A.1 Chord name chart ... 615
A.2 Common chord modifiers ... 616
A.3 Predefined string tunings ... 619
A.4 Predefined fretboard diagrams ... 621
Diagrams for Guitar ... 621
Diagrams for Ukulele ... 623
Diagrams for Mandolin ... 624
A.5 Predefined paper sizes ... 626
A.6 MIDI instruments ... 630
A.7 List of colors ... 630
A.8 The Feta font ... 632
Clef glyphs ... 632
Time Signature glyphs ... 632
Number glyphs ... 633
Accidental glyphs ... 633
Default Notehead glyphs ... 634
Special Notehead glyphs ... 634
Shape-note Notehead glyphs ... 635
Rest glyphs ... 639
Flag glyphs ... 640
Dot glyphs ... 640
Dynamic glyphs ... 640
Script glyphs ... 641
Arrowhead glyphs ... 643
Bracket-tip glyphs ... 643
Pedal glyphs ... 643
Accordion glyphs ... 644
Tie glyphs ... 644
Vaticana glyphs ... 644
Medicaea glyphs ... 645
Hufnagel glyphs ... 646
Mensural glyphs ... 646
Neomensural glyphs ... 650
Petrucci glyphs ... 651
Solesmes glyphs ... 652
Kievan Notation glyphs ... 652
A.9 Note head styles ... 653
A.10 Text markup commands ... 653
A.10.1 Font ... 653
A.10.2 Align ... 662
A.10.3 Graphic ... 677

A.10.4 Music ... 684
A.10.5 Instrument Specific Markup ... 689
A.10.6 Accordion Registers ... 693
A.10.7 Other ... 697
A.11 Text markup list commands ... 703
A.12 List of special characters ... 704
A.13 List of articulations ... 706
Articulation scripts ... 706
Ornament scripts ... 706
Fermata scripts ... 706
Instrument-specific scripts ... 707
Repeat sign scripts ... 707
Ancient scripts ... 707
A.14 Percussion notes ... 707
A.15 Technical glossary ... 709
alist ... 709
callback ... 709
closure ... 709
glyph ... 709
grob ... 709
immutable ... 710
interface ... 710
lexer ... 710
mutable ... 710
output-def ... 710
parser ... 710
parser variable ... 711
prob ... 711
simple closure ... 711
smob ... 711
stencil ... 711
A.16 All context properties ... 712
A.17 Layout properties ... 724
A.18 Available music functions ... 742
A.19 Context modification identifiers ... 751
A.20 Predefined type predicates ... 751
R5RS primary predicates ... 751
R5RS secondary predicates ... 752
Guile predicates ... 752
LilyPond scheme predicates ... 752
LilyPond exported predicates ... 753
A.21 Scheme functions ... 754

Appendice B Cheat sheet ... 778

Appendice C GNU Free Documentation License ... 782

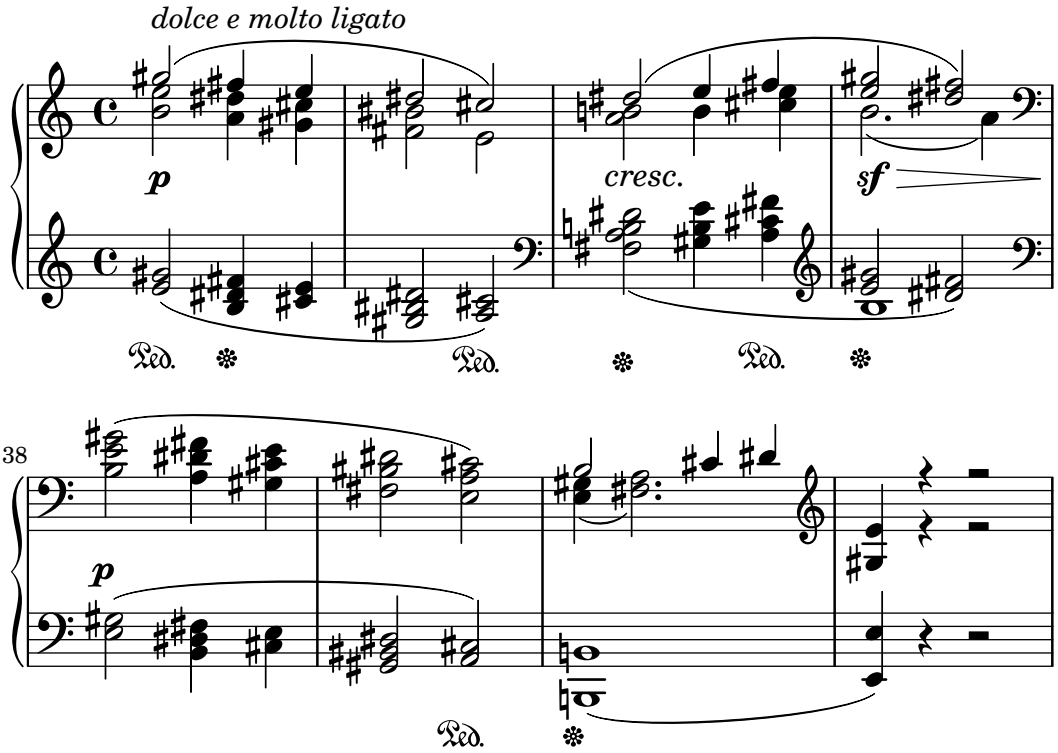
Appendice D Indice dei comandi di LilyPond ... 789

Appendice E Indice di LilyPond ... 799

1 Notazione musicale

Questo capitolo spiega come creare la notazione musicale.

1.1 Altezze



Questa sezione tratta il modo in cui si determina l'altezza delle note. Occorre considerare tre aspetti: input, modifica e output.
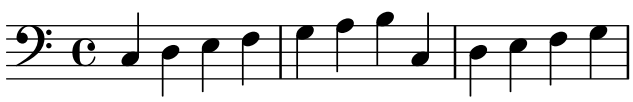
1.1.1 Inserimento delle altezze

Questa sezione spiega come indicare l'altezza delle note. Ci sono due modi di collocare le note in una determinata ottava: il modo assoluto e il modo relativo. Nella maggioranza dei casi il modo relativo è più funzionale.

Ottava assoluta
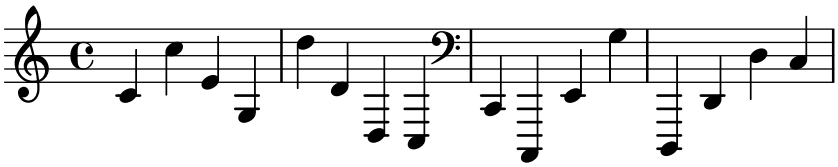
Le altezze, se non si adotta una lingua diversa, sono scritte in notazione olandese, che usa le lettere minuscole dalla `a` (La) alla `g` (Sol). Le note `c` (Do) e `b` (Si) vengono scritte un'ottava sotto il Do centrale.

```
{
  \clef bass
  c4 d e f
  g4 a b c
  d4 e f g
}
```



Si possono indicare altre ottave con l'apice singolo (`'`) o la virgola (`,`). Ogni `'` alza l'altezza di un'ottava; ogni `,` abbassa l'altezza di un'ottava.

```
{
  \clef treble
  c'4 c'' e' g
  d''4 d' d c
  \clef bass
  c,4 c,, e, g
  d,,4 d, d c
}
```



Si può indicare in modo esplicito che la musica viene inserita con l'ottava assoluta facendola precedere da `\absolute`:

`\absolute` *`espressione-musicale`*

verrà interpretata nella modalità assoluta indipendentemente dal contesto in cui si trova.

Vedi anche

Glossario musicale: Sezione "Nomi delle altezze" in *Glossario Musicale*.

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

Ottava relativa

L'inserimento delle note con l'ottava assoluta costringe a specificare l'ottava di ogni singola nota. Al contrario, se si usa l'ottava relativa, ogni ottava è determinata dall'ultima nota: se si cambia l'ottava di una nota, cambieranno anche le ottave di tutte le note successive.

La modalità relativa deve essere impostata in modo esplicito col comando `\relative`:

`\relative` *`altezza_di_riferimento espressione_musicale`*

In modalità relativa ogni nota è collocata il più vicino possibile a quella precedente. Questo significa che l'ottava di ogni altezza all'interno di *`espressione_musicale`* viene calcolata nel modo seguente:

- In assenza di segni di cambiamento d'ottava, l'ottava di un'altezza viene calcolata in modo che l'intervallo con la nota precedente sia inferiore a una quinta. Tale intervallo è determinato senza considerare gli accidenti.
- Si può aggiungere un segno di cambiamento d'ottava `'` o `,` per collocare l'altezza di una nota all'ottava superiore o inferiore a quella di riferimento.
- È possibile usare più di un segno di cambiamento d'ottava. Per esempio, `''` e `,,` modificano l'altezza di due ottave.
- L'altezza della prima nota è relativa a *`altezza_di_riferimento`*. *`altezza_di_riferimento`* è specificato nel modo di ottava assoluta. Quale di queste opzioni è la più conveniente?

  un'ottava del `c`
  : Identificare il Do centrale con `c'` è molto semplice, quindi trovare le ottave del `c` (Do) sarà altrettanto semplice. Se la musica inizia con `gis` sopra `c'''`, si scriverà qualcosa simile a `\relative c''' { gis' ... }`
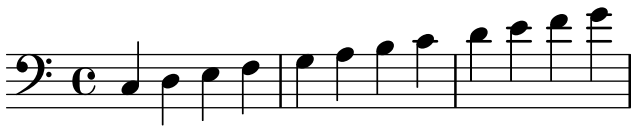
  un'ottava della prima nota
  : Scrivere `\relative gis''' { gis ... }` è un modo semplice per determinare l'altezza assoluta della prima nota dell'espressione musicale.

nessuna altezza di riferimento esplicita
: Questa (ovvero `\relative { gis''' ... }`) può essere considerata una versione compatta dell'opzione precedente: la prima nota dentro l'espressione musicale è scritta come altezza assoluta. In questo caso equivale a scegliere `f` come altezza di riferimento.

La documentazione di solito usa la prima opzione.
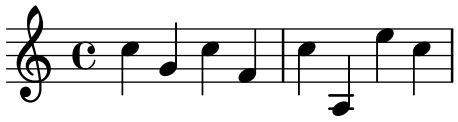
Ecco il modo relativo in azione:

```
\relative c {
  \clef bass
  c d e f
  g a b c
  d e f g
}
```
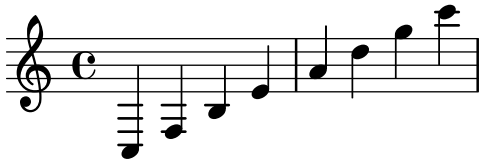


I segni di cambiamento d'ottava si impiegano per gli intervalli più ampi di quello di quarta:

```
\relative c'' {
  c g c f,
  c' a, e'' c
}
```
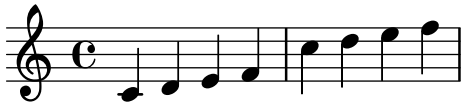


Una sequenza di note senza segni di ottava può tuttavia comprendere intervalli di grande estensione:

```
\relative c {
  c f b e
  a d g c
}
```



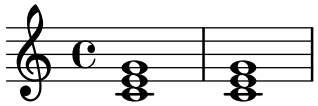
Nel caso di blocchi `\relative` annidati, si considera il blocco `\relative` più interno.

```
\relative c' {
  c d e f
  \relative c'' {
    c d e f
  }
}
```



`\relative` non ha effetto sui blocchi `\chordmode`.
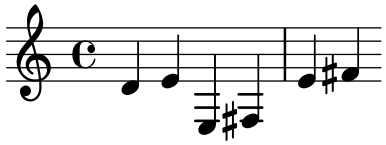
```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```



`\relative` non può essere inserito all'interno dei blocchi `\chordmode`.
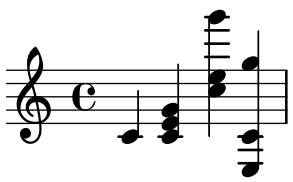
La musica all'interno di un blocco `\transpose` è considerata in notazione d'ottava assoluta, a meno che non sia incluso il blocco `\relative`.

```
\relative c' {
  d e
  \transpose f g {
    d e
    \relative c' {
      d e
    }
  }
}
```



Se l'elemento precedente è un accordo, il posizionamento dell'ottava della nota o dell'accordo che segue è riferito alla prima nota dell'accordo stesso. All'interno degli accordi la nota successiva è sempre relativa a quella precedente. Esaminate con attenzione l'esempio seguente, e in particolare le note `c`.
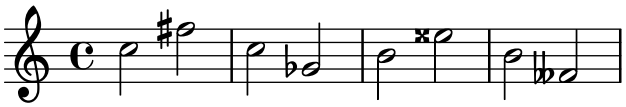
```
\relative c' {
  c
  <c e g>
  <c' e g'>
  <c, e, g''>
}
```



Come spiegato sopra, il riferimento delle altezze a un'ottava è calcolato in base ai soli nomi delle note, senza considerare le alterazioni. Dunque un Mi doppio diesis che segue un Si verrà posizionato sopra, mentre un Fa doppio bemolle sarà posizionato sotto. In altre parole, un

intervallo di quarta aumentata due volte viene considerato più piccolo di una quinta diminuita due volte, indipendentemente dal numero di semitoni contenuto in ogni intervallo.

```
\relative c'' {
  c2 fis
  c2 ges
  b2 eisis
  b2 feses
}
```



Ne consegue che la prima nota di un blocco `\relative f` venga interpretata come se fosse scritta nel modo di ottava assoluta.

Vedi anche

Glossario musicale: Sezione “quinta” in *Glossario Musicale*, Sezione “intervallo” in *Glossario Musicale*, Sezione “Nomi delle altezze” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Octave checks], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.
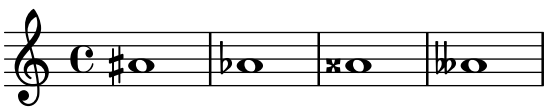
Guida al funzionamento interno: Sezione “RelativeOctaveMusic” in *Guida al Funzionamento Interno*.

Alterazioni

> **Nota:** I nuovi utenti sono talvolta confusi dalla gestione delle alterazioni e delle armature di chiave. In LilyPond i nomi delle note costituiscono l’input grezzo; le armature e le chiavi determinano come questo input grezzo venga mostrato. Una nota non alterata come `c` significa ‘Do naturale’, indipendentemente dall’armatura o dalla chiave. Per maggiori informazioni si veda Sezione “Alterazioni e armature di chiave” in *Manuale di Apprendimento*.
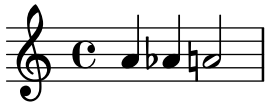
Nella modalità di notazione predefinita un *diesis* si ottiene aggiungendo `is` al nome della nota, un *bemolle* aggiungendo `es`. Come potete immaginare, un *doppio diesis* o *doppio bemolle* si ottengono aggiungendo `isis` o `eses`. Questa sintassi è desunta dalla notazione olandese. Per usare altri nomi per le alterazioni, si veda ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩.

```
ais1 aes aisis aeses
```



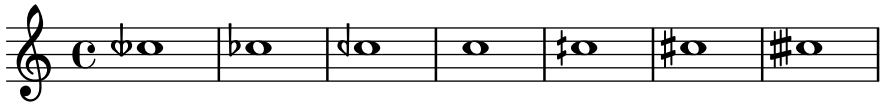
Un bequadro cancella l’effetto di un’alterazione o di un’armatura di chiave. Tuttavia, nella sintassi di Lilypond, non occorre specificare i bequadri mediante l’aggiunta di un particolare suffisso: un’altezza naturale è indicata con il semplice nome della nota:

```
a4 aes a2
```
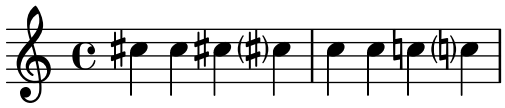


È possibile indicare alterazioni di quarti di tono. Ecco una serie di Do con altezza crescente:

```
ceseh1 ces ceh c cih cis cisih
```
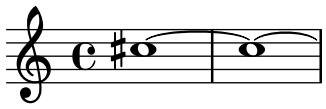


Di norma le alterazioni vengono mostrate automaticamente, ma è possibile anche inserirle manualmente. Si può forzare l'inserimento di un'alterazione di sicurezza aggiungendo il punto esclamativo `!` dopo l'altezza. Un'alterazione di cortesia (ovvero un'alterazione compresa tra parentesi) si ottiene aggiungendo il punto interrogativo `?` dopo l'altezza. Questi segni possono essere usati anche per produrre dei bequadri.
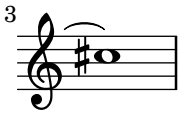
```
cis cis cis! cis? c c c! c?
```



Se una nota è prolungata attraverso una legatura di valore, l'alterazione viene ripetuta solo all'inizio di un nuovo sistema:

```
cis1~ cis~
\break
cis
```
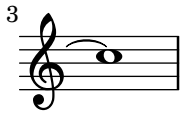




Frammenti di codice selezionati

*Nascondere le alterazioni delle note con legatura di valore*all'inizio di un nuovo sistema

Questo frammento mostra come nascondere le alterazioni delle note unite alla figura precedente mediante una legatura di valore all'inizio di un nuovo sistema

```
\relative c'' {
  \override Accidental.hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```
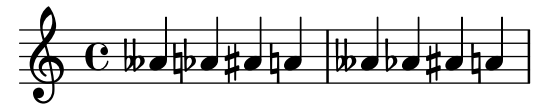




Impedire l'inserimento automatico dei bequadri supplementari

Secondo le norme tipografiche tradizionali, un segno di bequadro viene inserito prima di un diesis o di un bemolle se un precedente doppio diesis o bemolle sulla stessa nota è cancellato. Per cambiare questo comportamento e seguire la pratica contemporanea, si imposta la proprietà `extraNatural` su `f` (falso) nel contesto `Staff`.

```
\relative c'' {
  aeses4 aes ais a
  \set Staff.extraNatural = ##f
  aeses4 aes ais a
}
```



Vedi anche

Glossario musicale: Sezione "diesis" in *Glossario Musicale*, Sezione "bemolle" in *Glossario Musicale*, Sezione "doppio diesis" in *Glossario Musicale*, Sezione "doppio bemolle" in *Glossario Musicale*, Sezione "Nomi delle altezze" in *Glossario Musicale*, Sezione "quarto di tono" in *Glossario Musicale*.

Manuale di apprendimento: Sezione "Alterazioni e armature di chiave" in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Automatic accidentals], pagina ⟨undefined⟩, [Annotational accidentals (musica ficta)], pagina 422, ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩.

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Accidental_engraver" in *Guida al Funzionamento Interno*, Sezione "Accidental" in *Guida al Funzionamento Interno*, Sezione "AccidentalCautionary" in *Guida al Funzionamento Interno*, Sezione "accidental-interface" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti
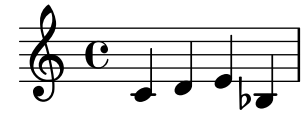
Poiché non esistono standard universalmente accettati per indicare le alterazioni di quarto di tono, il simbolo impiegato da LilyPond non si riferisce ad alcuno standard.

Nomi delle note in altre lingue

Lilypond comprende insiemi predefiniti di nomi di note e alterazioni in altre lingue. La scelta della lingua si fa solitamente all'inizio del file; l'esempio seguente è scritto in notazione italiana:

```
\language "italiano"

\relative do' {
  do re mi sib
}
```



Le lingue disponibili e i tipi di notazione che definiscono sono:
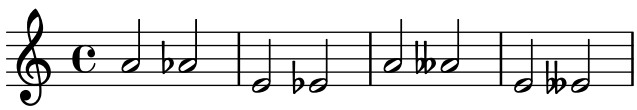
Lingua	**Nomi delle note**
`nederlands`	c d e f g a bes b
`catalan`	do re mi fa sol la sib si
`deutsch`	c d e f g a b h
`english`	c d e f g a bf b
`espanol` o `español`	do re mi fa sol la sib si
`italiano` o `français`	do re mi fa sol la sib si
`norsk`	c d e f g a b h
`portugues`	do re mi fa sol la sib si
`suomi`	c d e f g a b h
`svenska`	c d e f g a b h
`vlaams`	do re mi fa sol la sib si

Oltre ai nomi delle note, anche i suffissi per le alterazioni possono variare a seconda della lingua adottata:

Lingua	**diesis**	**bemolle**	**doppio diesis**	**doppio bemolle**
`nederlands`	-is	-es	-isis	-eses
`catalan`	-d/-s	-b	-dd/-ss	-bb
`deutsch`	-is	-es	-isis	-eses
`english`	-s/-sharp	-f/-flat	-ss/-x/-sharpsharp	-ff/-flatflat
`espanol` o `español`	-s	-b	-ss/-x	-bb
`italiano` o `français`	-d	-b	-dd	-bb
`norsk`	-iss/-is	-ess/-es	-ississ/-isis	-essess/-eses
`portugues`	-s	-b	-ss	-bb
`suomi`	-is	-es	-isis	-eses
`svenska`	-iss	-ess	-ississ	-essess
`vlaams`	-k	-b	-kk	-bb

In olandese, `aes` viene contratto in `as`, ma entrambe le forme sono accettate in LilyPond. Analogalmente, sia `es` che `ees` sono accettati. Lo stesso vale per `aeses` / `ases` e `eeses` / `eses`. Talvolta solo questi nomi contratti sono definiti nei corrispondenti file della lingua.

```
a2 as e es a ases e eses
```



In alcune forme musicali vengono usati i microtoni, le cui alterazioni sono frazioni di un 'normale' diesis o bemolle. La seguente tabella elenca i nomi delle note per le alterazioni di un quarto di tono in varie lingue; i prefissi *semi-* e *sesqui-* significano rispettivamente 'metà' e 'uno e mezzo'. Le lingue che non compaiono in questa tabella non hanno ancora dei nomi per le note speciali.

Lingua	semi-diesis	semi-bemolle	sesqui-diesis	sesqui-bemolle
`nederlands`	-ih	-eh	-isih	-eseh
`deutsch`	-ih	-eh	-isih	-eseh
`english`	-qs	-qf	-tqs	-tqf
`espanol` o `español`	-cs	-cb	-tcs	-tcb
`italiano` o `français`	-sd	-sb	-dsd	-bsb
`portugues`	-sqt	-bqt	-stqt	-btqt

Gran parte delle lingue presentate qui sono comunemente associate alla musica classica occidentale, nota anche come *Common Practice Period*. Sono tuttavia supportati anche altezze e sistemi di accordatura alternativi: si veda Sezione 2.10.1 [Common notation for non-Western music], pagina 443.

Vedi anche

Glossario musicale: Sezione “Nomi delle altezze” in *Glossario Musicale*, Sezione “Periodo di pratica comune” in *Glossario Musicale*.

Guida alla notazione: Sezione 2.10.1 [Common notation for non-Western music], pagina 443.

File installati: ‘`scm/define-note-names.scm`’.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

1.1.2 Modifica di più altezze
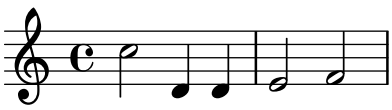
Questa sezione tratta il modo di modificare le altezze delle note.

Controlli di ottava

In modalità relativa è facile dimenticare un segno di cambiamento d’ottava. I controlli di ottava permettono di rilevare questi errori più facilmente: infatti, generano un avviso e correggono l’ottava se una nota si trova in un’ottava diversa dal previsto.

Per controllare l’ottava di una nota, occorre specificare l’ottava assoluta dopo il simbolo `=`. Questo esempio genererà un avviso (e cambierà l’altezza) perché la seconda nota è l’ottava assoluta `d''` invece di `d'`, come indicato dalla correzione di ottava.
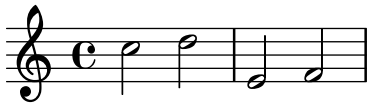
```
\relative c'' {
  c2 d='4 d
  e2 f
}
```



L’ottava in cui si trovano le note può essere controllata anche col comando `\octaveCheck` *`altezza_di_controllo`*. L’*`altezza_di_controllo`* è specificata in modo assoluto. Questo comando controlla che l’intervallo tra la nota precedente e l’*`altezza_di_controllo`* sia compresa in una quinta (ovvero secondo il normale calcolo della modalità relativo). Se il controllo fallisce, compare un avviso, ma la nota precedente non viene modificata. Le note successive sono relative all’*`altezza_di_controllo`*.

```
\relative c'' {
  c2 d
```
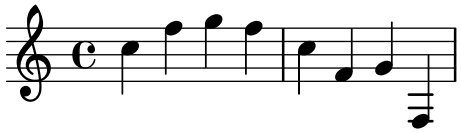
```
  \octaveCheck c'
  e2 f
}
```



Nelle due battute che seguono, il primo e il terzo `\octaveCheck` falliscono, mentre il secondo non fallisce.

```
\relative c'' {
  c4 f g f

  c4
  \octaveCheck c'
  f
  \octaveCheck c'
  g
  \octaveCheck c'
  f
}
```



Vedi anche

Frammenti di codice: Sezione “Pitches, Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “RelativeOctaveCheck” in *Guida al Funzionamento Interno*.

Trasposizione

Un’espressione musicale può essere trasposta con `\transpose`. La sintassi è

`\transpose` *`altezza_di_partenza altezza_di_arrivo espressione_musicale`*
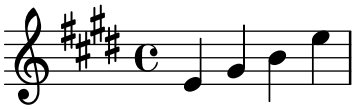
Significa che *`espressione_musicale`* viene trasposto dell’intervallo compreso tra le altezze *`altezza_di_partenza`* e *`altezza_di_arrivo`*: qualsiasi nota che presenti un’altezza corrispondente all’*`altezza_di_partenza`* viene modificata in *`altezza_di_arrivo`*, e qualsiasi altra nota viene trasposta dello stesso intervallo. Entrambe le altezze sono inserite in modalità assoluta.

> **Nota:** La musica all’interno di un blocco `\transpose` è assoluta a meno che il blocco non includa un `\relative`.

Prendiamo come esempio un brano scritto in Re maggiore. Possiamo trasportarlo in Mi maggiore; si noti come anche l’armatura di chiave venga trasposta automaticamente.
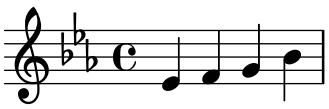
```
\transpose d e {
  \relative c' {
    \key d \major
    d4 fis a d
  }
```

```
}
```



Se una parte scritta in Do (l'*intonazione reale* abituale) deve essere suonata su un clarinetto in La (per il quale un La viene rappresentato da un Do e dunque suona una terza minore più basso), la trasposizione sarà ottenuta con:
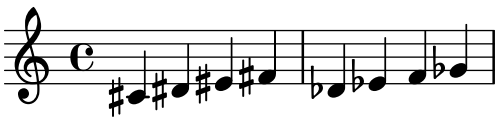
```
\transpose a c' {
  \relative c' {
    \key c \major
    c4 d e g
  }
}
```



Si noti che `\key c \major` è specificato esplicitamente. Se non si specifica un'armatura di chiave, le note verranno trasposte ma non apparirà alcuna armatura.

`\transpose` fa distinzione tra altezze enarmoniche: sia `\transpose c cis` che `\transpose c des` traspongono un brano di un semitono più alto. La prima versione mostrerà i diesis e le note rimarranno sullo stesso grado della scala, mentre la seconda versione mostrerà i bemolle sul grado superiore della scala.

```
music = \relative c' { c d e f }
\new Staff {
  \transpose c cis { \music }
  \transpose c des { \music }
}
```



`\transpose` può essere usato anche in un altro modo, ovvero per inserire note scritte per uno strumento traspositore. Gli esempi precedenti mostrano come inserire altezze in Do (o *intonazione reale*) e mostrare le note di uno strumento traspositore, ma è possibile anche il contrario: per esempio, se da un insieme di parti strumentali si volesse ricavare una partitura per il direttore. Così, per inserire la parte per una tromba in Si bemolle che inizia con un Mi (intonazione reale Re), si può scrivere:

```
musicInBflat = { e4 ... }
\transpose c bes, \musicInBflat
```

Per stampare questa musica in Fa (ad esempio per riarrangiarla per corno) si può avvolgere la musica esistente in un altro `\transpose`:

```
musicInBflat = { e4 ... }
\transpose f c' { \transpose c bes, \musicInBflat }
```

Per maggiori informazioni sugli strumenti traspositori, si veda ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩.

Frammenti di codice selezionati

Trasposizione delle altezze con numero minimo di alterazioni

Questo esempio usa del codice Scheme per imporre delle modifiche enarmoniche alle note che permettano di avere il numero minimo di alterazioni. In questo caso si applica la seguente regola:

Le doppie alterazioni devono essere eliminate

Si diesis -> Do

Mi diesis -> Fa

Do bemolle -> Si

Fa bemolle -> Mi

In questo modo vengono scelti i suoni enarmonici più semplici.

```
  #(define (naturalize-pitch p)
     (let ((o (ly:pitch-octave p))
           (a (* 4 (ly:pitch-alteration p)))
           ;; alteration, a, in quarter tone steps,
           ;; for historical reasons
           (n (ly:pitch-notename p)))
       (cond
        ((and (> a 1) (or (eq? n 6) (eq? n 2)))
         (set! a (- a 2))
         (set! n (+ n 1)))
        ((and (< a -1) (or (eq? n 0) (eq? n 3)))
         (set! a (+ a 2))
         (set! n (- n 1))))
       (cond
        ((> a 2) (set! a (- a 4)) (set! n (+ n 1)))
        ((< a -2) (set! a (+ a 4)) (set! n (- n 1))))
       (if (< n 0) (begin (set! o (- o 1)) (set! n (+ n 7))))
       (if (> n 6) (begin (set! o (+ o 1)) (set! n (- n 7))))
       (ly:make-pitch o n (/ a 4))))
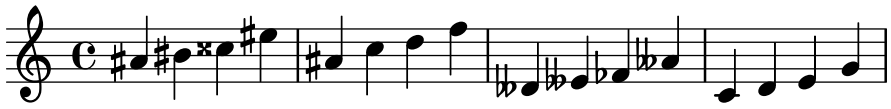
  #(define (naturalize music)
     (let ((es (ly:music-property music 'elements))
           (e (ly:music-property music 'element))
           (p (ly:music-property music 'pitch)))
       (if (pair? es)
           (ly:music-set-property!
            music 'elements
            (map (lambda (x) (naturalize x)) es)))
       (if (ly:music? e)
           (ly:music-set-property!
            music 'element
            (naturalize e)))
       (if (ly:pitch? p)
           (begin
             (set! p (naturalize-pitch p))
             (ly:music-set-property! music 'pitch p)))
       music))

  naturalizeMusic =
```

```
#(define-music-function (parser location m)
   (ly:music?)
   (naturalize m))

music = \relative c' { c4 d e g }

\score {
  \new Staff {
    \transpose c ais { \music }
    \naturalizeMusic \transpose c ais { \music }
    \transpose c deses { \music }
    \naturalizeMusic \transpose c deses { \music }
  }
  \layout { }
}
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩, ⟨undefined⟩ [Inversion], pagina ⟨undefined⟩, ⟨undefined⟩ [Modal transformations], pagina ⟨undefined⟩, ⟨undefined⟩ [Relative octave entry], pagina ⟨undefined⟩, ⟨undefined⟩ [Retrograde], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TransposedMusic” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

La conversione relativa non avrà effetto sulle sezioni `\transpose`, `\chordmode` e `\relative` comprese all’interno di un blocco `\relative`. Per usare la modalità relativa all’interno di musica trasposta, occorre inserire un ulteriore blocco `\relative` all’interno di `\transpose`.

Il comando `\transpose` impedisce di stampare le alterazioni triple. Le sostituisce con un’altezza ‘enarmonicamente equivalente’ (per esempio, Re bemolle al posto di Mi triplo bemolle).
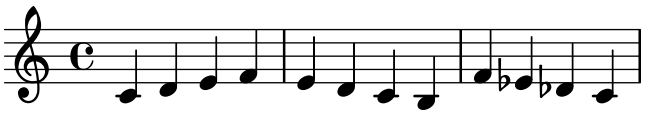
Inversione

Un’espressione musicale può essere invertita e trasposta in una singola operazione con:

`\inversion` *`altezza-di-riferimento altezza-di-arrivo espressione_musicale`*

L’*`espressione_musicale`* viene invertita intervallo per intervallo intorno all’*`altezza-di-riferimento`* e poi trasposta in modo che ci sia una corrispondenza tra *`altezza-di-riferimento`* e *`altezza-di-arrivo`*.

```
music = \relative c' { c d e f }
\new Staff {
  \music
  \inversion d' d' \music
  \inversion d' ees' \music
}
```



Nota: I motivi da invertire devono essere scritti in forma assoluta oppure devono essere prima convertiti in forma assoluta racchiudendoli in un blocco `\relative`.

Vedi anche
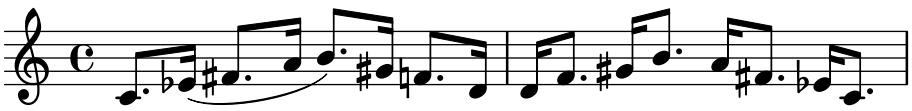
Guida alla notazione: ⟨undefined⟩ [Modal transformations], pagina ⟨undefined⟩, ⟨undefined⟩ [Retrograde], pagina ⟨undefined⟩, ⟨undefined⟩ [Transpose], pagina ⟨undefined⟩.

Retrogradazione

Un'espressione musicale può essere invertita in modo da produrre il proprio retrogrado:

```
music = \relative c' { c8. ees16( fis8. a16 b8.) gis16 f8. d16 }

\new Staff {
  \music
  \retrograde \music
}
```



Problemi noti e avvertimenti

Le legature di valore manuali in `\retrograde` saranno spezzate e genereranno degli avvisi. Alcune legature di valore possono essere generate automaticamente abilitando ⟨undefined⟩ [Automatic note splitting], pagina ⟨undefined⟩.

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Inversion], pagina ⟨undefined⟩, ⟨undefined⟩ [Modal transformations], pagina ⟨undefined⟩, ⟨undefined⟩ [Transpose], pagina ⟨undefined⟩.

Trasposizioni modali

In una composizione musicale basata su una scala, un motivo viene frequentemente trasportato in differenti modi. Può essere *trasposto* per iniziare in punti diversi della scala o può essere *invertito* rispetto a un punto cardine della scala. Può anche essere rovesciato per produrre il *retrogrado*, si veda ⟨undefined⟩ [Retrograde], pagina ⟨undefined⟩.

Nota: Le note che non si trovano all'interno della scala definita non vengono trasformate.

Trasposizione modale
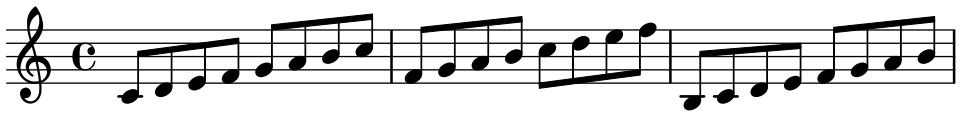
Un motivo può essere trasposto entro una certa scala con:

```
\modalTranspose altezza-di-partenza altezza-di-arrivo scala motif
```

Le note di *motif* vengono spostate, se all'interno della *scala*, del numero di gradi della scala dati dall'intervallo tra *altezza-di-arrivo* e *altezza-di-partenza*:

```
diatonicScale = \relative c' { c d e f g a b }
motif = \relative c' { c8 d e f g a b c }
```
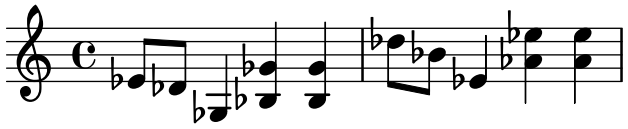
```
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
```



È possibile indicare una scala ascendente di qualsiasi lunghezza e con qualsiasi intervallo:

```
pentatonicScale = \relative c' { ges aes bes des ees }
motif = \relative c' { ees8 des ges,4 <ges' bes,> <ges bes,> }
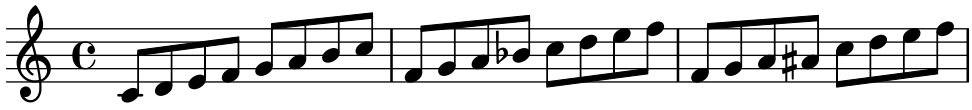
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
```



Se usato con una scala cromatica, `\modalTranspose` ha un effetto simile a `\transpose`, con in più la possibilità di specificare i nomi delle note da usare:

```
chromaticScale = \relative c' { c cis d dis e f fis g gis a ais b }
motif = \relative c' { c8 d e f g a b c }

\new Staff {
  \motif
  \transpose c f \motif
  \modalTranspose c f \chromaticScale \motif
}
```



Inversione modale

Una sequenza di note può essere invertita all'interno di una data scala intorno a una determinata nota cardine e quindi trasposto, in un'unica operazione, con:
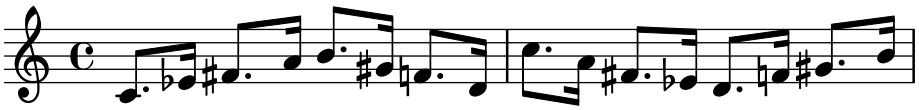
`\modalInversion` *`altezza-cardine altezza-di-arrivo scala motif`*

Le note di *motif* vengono spostate dello stesso numero di gradi dalla nota dell'*altezza-cardine* all'interno della *scala*, ma nella direzione opposta, e il risultato viene poi spostato all'interno della *scala* per il numero di gradi dato dall'intervallo tra *altezza-di-arrivo* e *altezza-cardine*.

Dunque, per invertire intorno a una particolare nota della scala, è necessario usare il medesimo valore per *altezza-cardine* e *altezza-di-arrivo*:

```
octatonicScale = \relative c' { ees f fis gis a b c d }
motif = \relative c' { c8. ees16 fis8. a16 b8. gis16 f8. d16 }
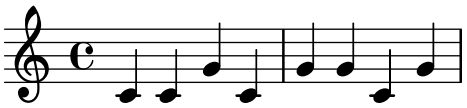
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
```



Per invertire intorno a una nota cardine posta tra altre due note, si inverte intorno a una della note e poi si traspone di un grado della scala. Le due note specificate possono essere interpretate come parentesi del punto cardine:

```
scale = \relative c' { c g' }
motive = \relative c' { c c g' c, }
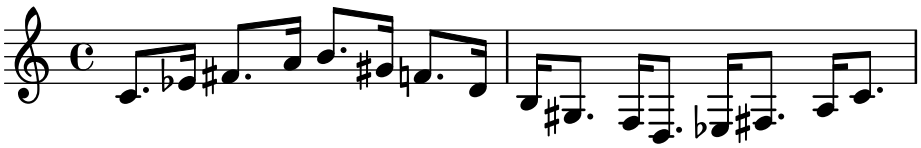
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
```



L'operazione combinata di inversione e retrogradazione produce la retrogradazione inversa:

```
octatonicScale = \relative c' { ees f fis gis a b c d }
motif = \relative c' { c8. ees16 fis8. a16 b8. gis16 f8. d16 }

\new Staff {
  \motif
  \retrograde \modalInversion c' c' \octatonicScale \motif
}
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Inversion], pagina ⟨undefined⟩, ⟨undefined⟩ [Retrograde], pagina ⟨undefined⟩, ⟨undefined⟩ [Transpose], pagina ⟨undefined⟩.
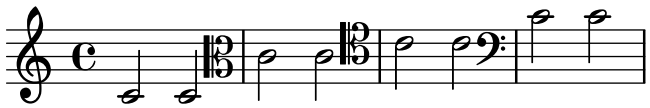
1.1.3 Aspetto delle altezze

Questa sezione tratta il modo di modificare l'aspetto delle altezze delle note.

Chiave

È possibile cambiare la chiave impiegata. Negli esempi seguenti mostriamo il Do centrale. I seguenti nomi di chiave possono (ma non devono) essere racchiusi tra virgolette.

```
\clef treble
c2 c
\clef alto
c2 c
\clef tenor
c2 c
\clef bass
c2 c
```



Altre chiavi:

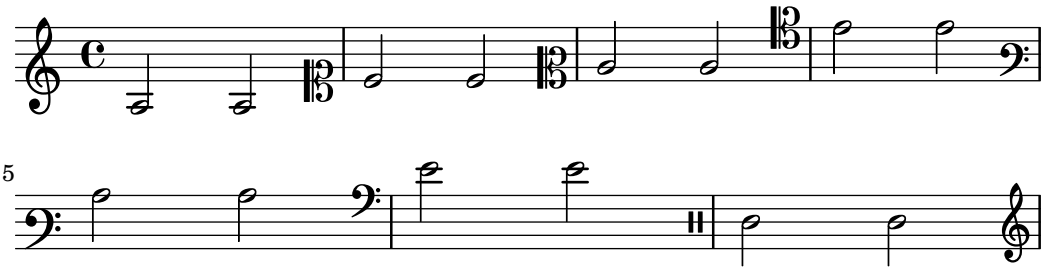
```
\clef french
c2 c
\clef soprano
c2 c
\clef mezzosoprano
c2 c
\clef baritone
c2 c
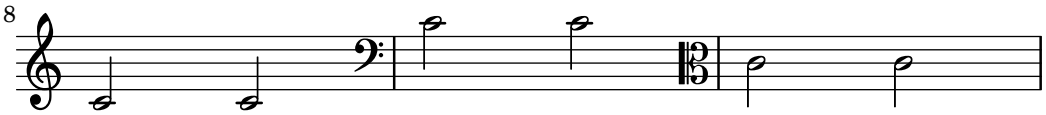
\break

\clef varbaritone
c2 c
\clef subbass
c2 c
\clef percussion
c2 c

\break
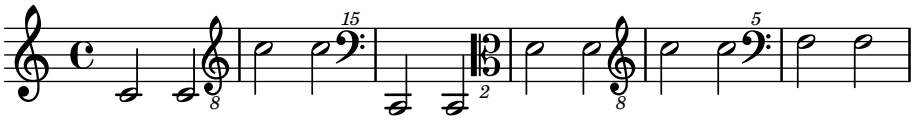
\clef G   % synonym for treble
c2 c
\clef F   % synonym for bass
c2 c
\clef C   % synonym for alto
c2 c
```
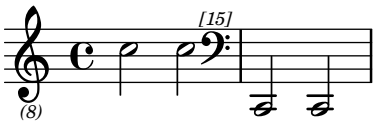




Aggiungendo `_8` o `^8` al nome della chiave, la sua adozione comporta il trasporto all'ottava rispettivamente inferiore o superiore, mentre `_15` e `^15` traspongono di due ottave. È possibile usare altri numeri interi, se necessario. I nomi di chiave contenenti caratteri non alfabetici devono essere racchiusi tra virgolette

```
\clef treble
c2 c
\clef "treble_8"
c2 c
\clef "bass^15"
c2 c
\clef "alto_2"
c2 c
\clef "G_8"
c2 c
\clef "F^5"
c2 c
```



L'ottavazione opzionale si può ottenere racchiudendo l'argomento numerico tra parentesi tonde o quadre:

```
\clef "treble_(8)"
c2 c
\clef "bass^[15]"
c2 c
```



Le altezze vengono mostrate come se l'argomento numerico fosse inserito senza parentesi.

Alcune chiavi particolari sono descritte in [Mensural clefs], pagina 417, [Gregorian clefs], pagina 425, [Default tablatures], pagina 326 e [Custom tablatures], pagina 339. Per alternare chiavi diverse nelle citazioni in corpo più piccolo all'interno di una partitura, si vedano le funzioni `\cueClef` e `\cueDuringWithClef` in ⟨undefined⟩ [Formatting cue notes], pagina ⟨undefined⟩.

Frammenti di codice selezionati

Modifiche manuali della proprietà della chiave

Il comando `\clef "treble_8"` equivale a impostare `clefGlyph`, `clefPosition` (che regola la posizione verticale della chiave), `middleCPosition` e `clefTransposition`. Viene stampata una chiave quando cambia una di queste proprietà, eccetto `middleCPosition`.

La modifica del glifo, della posizione della chiave o dell'ottavazione non è sufficiente per cambiare la posizione delle note che seguono sul rigo: bisogna anche specificare la posizione del Do centrale (middle C). Per far sì che le armature di chiave si trovino sulle linee corrette del rigo, occorre impostare anche `middleCClefPosition`. I parametri di posizione sono relativi alla linea centrale del rigo, con i numeri positivi che indicano la parte superiore: ogni linea e spazio

valgono uno. Il valore `clefTransposition` di norma è impostato su 7, -7, 15 o -15, ma altri valori sono considerati validi.

Quando un cambio di chiave avviene in corrispondenza di un'interruzione di linea, di norma il simbolo della nuova chiave viene inserito sia alla fine del rigo precedente sia all'inizio di quello successivo. Se la chiave di avvertimento a fine rigo non fosse necessaria, può essere nascosta impostando la proprietà `explicitClefVisibility` del contesto `Staff` su `end-of-line-invisible`. Il comportamento predefinito può essere ripristinato con `\unset Staff.explicitClefVisibility`.

Gli esempi seguenti mostrano le possibilità date dall'impostazione manuale di tali proprietà. Sulla prima linea le modifiche manuali preservano il posizionamento relativo standard di chiavi e note, mentre sulla seconda linea non lo fanno.

```
\layout { ragged-right = ##t }
{
  % The default treble clef
  \key f \major
  c'1
  % The standard bass clef
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  \set Staff.middleCPosition = #6
  \set Staff.middleCClefPosition = #6
  \key g \major
  c'1
  % The baritone clef
  \set Staff.clefGlyph = #"clefs.C"
  \set Staff.clefPosition = #4
  \set Staff.middleCPosition = #4
  \set Staff.middleCClefPosition = #4
  \key f \major
  c'1
  % The standard choral tenor clef
  \set Staff.clefGlyph = #"clefs.G"
  \set Staff.clefPosition = #-2
  \set Staff.clefTransposition = #-7
  \set Staff.middleCPosition = #1
  \set Staff.middleCClefPosition = #1
  \key f \major
  c'1
  % A non-standard clef
  \set Staff.clefPosition = #0
  \set Staff.clefTransposition = #0
  \set Staff.middleCPosition = #-4
  \set Staff.middleCClefPosition = #-4
  \key g \major
  c'1 \break

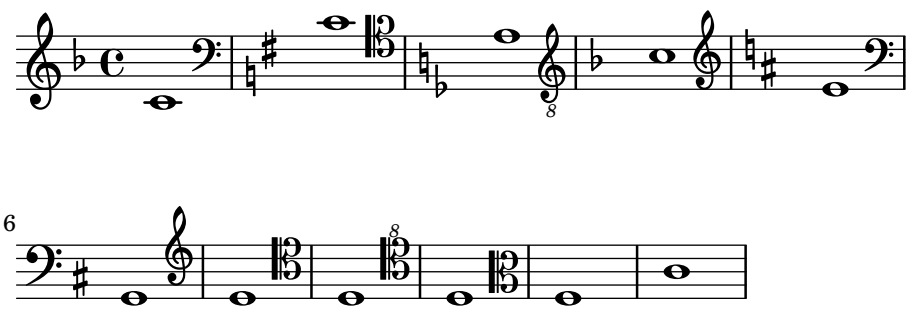
  % The following clef changes do not preserve
  % the normal relationship between notes, key signatures
  % and clefs:

  \set Staff.clefGlyph = #"clefs.F"
```

```
  \set Staff.clefPosition = #2
  c'1
  \set Staff.clefGlyph = #"clefs.G"
  c'1
  \set Staff.clefGlyph = #"clefs.C"
  c'1
  \set Staff.clefTransposition = #7
  c'1
  \set Staff.clefTransposition = #0
  \set Staff.clefPosition = #0
  c'1

  % Return to the normal clef:

  \set Staff.middleCPosition = #0
  c'1
}
```



Vedi anche
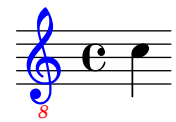
Guida alla notazione: [Mensural clefs], pagina 417, [Gregorian clefs], pagina 425, [Default tablatures], pagina 326, [Custom tablatures], pagina 339, ⟨undefined⟩ [Formatting cue notes], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Clef_engraver” in *Guida al Funzionamento Interno*, Sezione “Clef” in *Guida al Funzionamento Interno*, Sezione “ClefModifier” in *Guida al Funzionamento Interno*, Sezione “clef-interface” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

I numeri di ottavazione assegnati alle chiavi sono trattati come oggetti grafici separati. Quindi qualsiasi `\override` all’oggetto *Clef* dovrà essere applicato, con un altro `\override`, all’oggetto *ClefModifier*.



Armatura di chiave
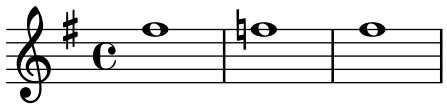
> **Nota:** I nuovi utenti sono talvolta confusi dalla gestione delle alterazioni e delle armature di chiave. In LilyPond i nomi delle note costituiscono l'input grezzo; le armature e le chiavi determinano come questo venga mostrato. Una nota non alterata come `c` significa 'Do naturale', indipendentemente dall'armatura o dalla chiave. Per maggiori informazioni si veda Sezione "Alterazioni e armature di chiave" in *Manuale di Apprendimento*.

L'armatura di chiave indica la tonalità di un brano. È costituita da un insieme di alterazioni (bemolle o diesis) all'inizio del rigo. L'armatura di chiave può essere modificata:

```
\key altezza modo
```

modo deve essere `\major` o `\minor` per ottenere rispettivamente un'armatura di *altezza*-maggiore o *altezza*-minore. È anche possibile usare i nomi tradizionali dei modi, chiamati anche *modi ecclesiastici*: `\ionian`, `\dorian`, `\phrygian`, `\lydian`, `\mixolydian`, `\aeolian` e `\locrian`.
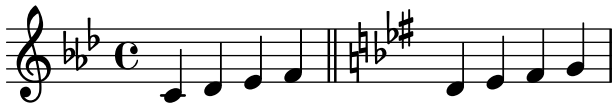
```
\key g \major
fis1
f
fis
```



Si possono definire ulteriori modi elencando le alterazioni per ogni grado della scala quando il modo inizia col Do.

```
freygish = #`((0 . ,NATURAL) (1 . ,FLAT) (2 . ,NATURAL)
    (3 . ,NATURAL) (4 . ,NATURAL) (5 . ,FLAT) (6 . ,FLAT))
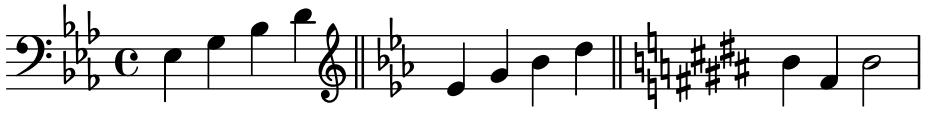
\relative c' {
  \key c \freygish c4 des e f
  \bar "||" \key d \freygish d es fis g
}
```



Le alterazioni dell'armatura di chiave possono essere collocate in posizioni diverse da quelle tradizionali o anche in più di un'ottava, usando le proprietà `flat-positions` e `sharp-positions` di `KeySignature`. I valori di queste proprietà specificano l'estensione delle posizioni del rigo in cui potranno comparire le alterazioni. Se viene specificata una sola posizione, le alterazioni vengono collocate entro l'ottava che finisce in quella posizione del rigo.

```
\override Staff.KeySignature.flat-positions = #'((-5 . 5))
\override Staff.KeyCancellation.flat-positions = #'((-5 . 5))
\clef bass \key es \major es g bes d
\clef treble \bar "||" \key es \major es g bes d

\override Staff.KeySignature.sharp-positions = #'(2)
\bar "||" \key b \major b fis b2
```
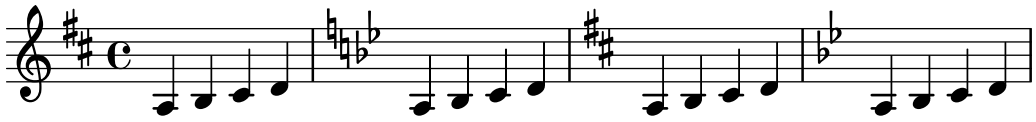


Frammenti di codice selezionati

Impedire l'inserimento dei segni di bequadro quando cambia l'armatura di chiave

Quando l'armatura di chiave cambia, vengono inseriti automaticamente i segni di bequadro per annulare le alterazioni di precedenti armature. Si può evitare questo comportamento impostando su `f` (falso) la proprietà `printKeyCancellation` nel contesto `Staff`.

```
\relative c' {
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
  \set Staff.printKeyCancellation = ##f
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
}
```



Armature di chiave non tradizionali

Il comando `\key` comunemente usato imposta la proprietà `keySignature`, che fa parte del contesto `Staff`.
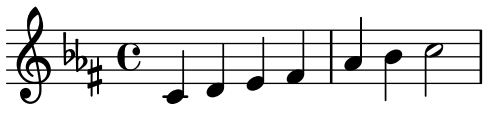
Per creare armature di chiave non standard, tale proprietà va impostata esplicitamente. Il formato di questo comando è una lista:

`\set Staff.keySignature = #`(((ottava . grado) . alterazione) ((ottava . grado) . alterazione) ...)` dove, per ogni elemento della lista, `ottava` indica l'ottava (0 è l'ottava dal Do centrale al Si precedente), `grado` indica la nota all'interno dell'ottava (0 significa Do e 6 significa Si) e `alterazione` può essere `,SHARP ,FLAT ,DOUBLE-SHARP` etc. (Si noti la virgola iniziale.)

Altrimenti, usando, per ogni elemento della lista, il formato breve `(grado . alterazione)`, ciò indica che la stessa alterazione deve essere presente in tutte le ottave.

Ecco un esempio di una possibile armatura per generare una scala a tono intero:

```
\relative c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
                              ((0 . 5) . ,FLAT)
                              ((0 . 3) . ,SHARP))
  c4 d e fis
  aes4 bes c2
}
```



Vedi anche

Glossario musicale: Sezione “church mode” in *Glossario Musicale*, Sezione “scordatura” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Alterazioni e armature di chiave” in *Manuale di Apprendimento*.
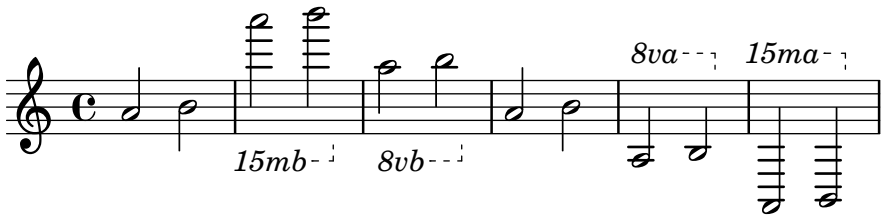
Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “KeyChangeEvent” in *Guida al Funzionamento Interno*, Sezione “Key_engraver” in *Guida al Funzionamento Interno*, Sezione “Key_performer” in *Guida al Funzionamento Interno*, Sezione “KeyCancellation” in *Guida al Funzionamento Interno*, Sezione “KeySignature” in *Guida al Funzionamento Interno*, Sezione “key-signature-interface” in *Guida al Funzionamento Interno*.

Segni di ottavazione

I *segni di ottavazione* introducono un’ulteriore trasposizione di ottava nel rigo:

```
a2 b
\ottava #-2
a2 b
\ottava #-1
a2 b
\ottava #0
a2 b
\ottava #1
a2 b
\ottava #2
a2 b
```

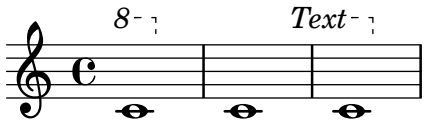


Frammenti di codice selezionati

Testo dell’ottava

Internamente, `\ottava` imposta le proprietà `ottavation` (ad esempio, su `8va` o `8vb`) e `middleCPosition`. Per sovrascrivere il testo della parentesi, occorre specificare `ottavation` dopo il comando `\ottava`.

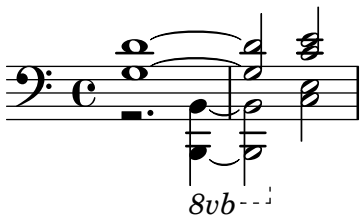
```
{
  \ottava #1
  \set Staff.ottavation = #"8"
  c''1
  \ottava #0
  c'1
  \ottava #1
  \set Staff.ottavation = #"Text"
  c''1
}
```



Aggiungere un segno di ottava a una sola voce

Se il rigo ha più di una voce, l'ottavazione in una voce trasporrà la posizione delle note in tutte le voci per la durata della parentesi dell'ottava. Se si intende applicare l'ottavazione a una sola voce, si possono impostare esplicitamente middleCPosition e la parentesi di ottava. In questo frammento, la chiave di basso ha di norma il MiddleCPosition impostato su 6, ovvero sei posizioni sopra la linea centrale, dunque nella porzione con l'ottava il MiddleCPosition è più alto di sette posizioni (un'ottava).

```
{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
    {
      r2.
      \set Staff.ottavation = #"8vb"
      \once \override Staff.OttavaBracket.direction = #DOWN
      \set Voice.middleCPosition = #(+ 6 7)
      <b,,, b,,>4 ~ |
      q2
      \unset Staff.ottavation
      \unset Voice.middleCPosition
      <c e>2
    }
  >>
}
```



Vedi anche

Glossario musicale: Sezione "ottavazione" in *Glossario Musicale*.

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Ottava_spanner_engraver" in *Guida al Funzionamento Interno*, Sezione "OttavaBracket" in *Guida al Funzionamento Interno*, Sezione "ottava-bracket-interface" in *Guida al Funzionamento Interno*.

Trasporto strumentale

Quando si scrivono partiture che comprendono strumenti traspositori, alcune parti possono essere scritte a un'altezza diversa dall'*intonazione reale*. In questi casi, è necessario specificare la chiave dello *strumento traspositore*, altrimenti l'output MIDI e le citazioni in altre parti produrranno altezze errate. Per maggiori informazioni sulle citazioni, si veda ⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩.

`\transposition` *altezza*

L'altezza da usare per `\transposition` deve corrispondere al suono effettivamente prodotto quando un `c'` scritto sul rigo viene suonato dallo strumento traspositore. Tale altezza viene
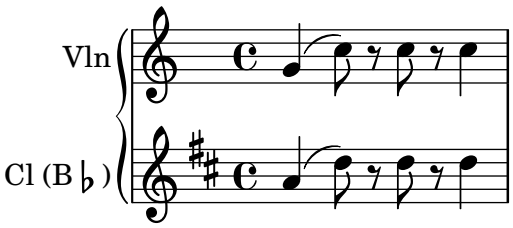
inserita in modalità assoluta; dunque, uno strumento che produce un suono reale un tono sopra la notazione deve usare `\transposition d'`. `\transposition` va usato *soltanto* se le altezze *non* sono scritte in intonazione reale.

Ecco un frammento per violino e clarinetto in Si bemolle, le cui parti sono inserite usando le note e l'armatura di chiave che appaiono nei rispettivi righi sulla partitura del direttore. I due strumenti suonano all'unisono.

```
\new GrandStaff <<
  \new Staff = "violin" {
    \relative c'' {
      \set Staff.instrumentName = #"Vln"
      \set Staff.midiInstrument = #"violin"
      % not strictly necessary, but a good reminder
      \transposition c'

      \key c \major
      g4( c8) r c r c4
    }
  }
  \new Staff = "clarinet" {
    \relative c'' {
      \set Staff.instrumentName = \markup { Cl (B\flat) }
      \set Staff.midiInstrument = #"clarinet"
      \transposition bes

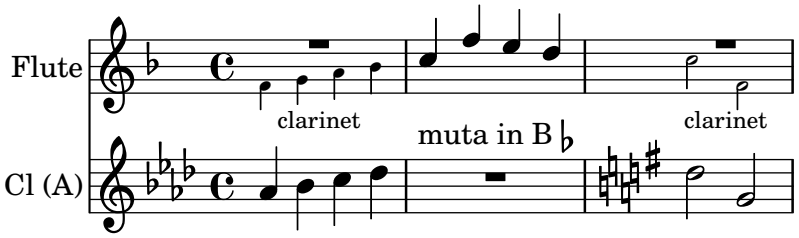
      \key d \major
      a4( d8) r d r d4
    }
  }
>>
```



`\transposition` può essere modificato nel corso di un brano. Ad esempio, un clarinettista potrebbe essere costretto a passare da un clarinetto in La a uno in Si bemolle.

```
flute = \relative c'' {
  \key f \major
  \cueDuring #"clarinet" #DOWN {
    R1 _\markup\tiny "clarinet"
    c4 f e d
    R1 _\markup\tiny "clarinet"
  }
}
clarinet = \relative c'' {
  \key aes \major
  \transposition a
  aes4 bes c des
```

```
  R1^\markup { muta in B\flat }
  \key g \major
  \transposition bes
  d2 g,
}
\addQuote "clarinet" \clarinet
<<
  \new Staff \with { instrumentName = #"Flute" }
    \flute
  \new Staff \with { instrumentName = #"Cl (A)" }
    \clarinet
>>
```



Vedi anche

Glossario musicale: Sezione “intonazione reale” in *Glossario Musicale*, Sezione “strumento traspositore” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩, ⟨undefined⟩ [Transpose], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Alterazioni automatiche

Esistono diverse convenzioni sul modo di scrivere le alterazioni. LilyPond ha una funzione per specificare lo stile di gestione delle alterazioni adottato. Questa funzione viene richiamata nel modo seguente:

```
\new Staff <<
  \accidentalStyle voice
  { ... }
>>
```

La gestione delle alterazioni si applica di norma all’attuale `Staff` (con l’eccezione degli stili `piano` e `piano-cautionary`, che sono spiegati dopo). Questa funzione accetta un secondo argomento opzionale che determina in quale ambito debba essere cambiato lo stile. Ad esempio, per usare lo stesso stile in tutti i righi dell’attuale `StaffGroup`, si usa:

```
\accidentalStyle StaffGroup.voice
```

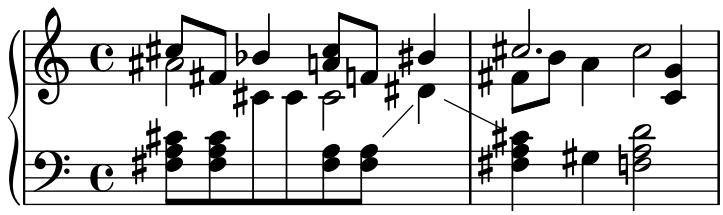
Sono supportati i seguenti modi di gestire le alterazioni. Il seguente esempio mostra tutti gli stili:

```
musicA = {
  <<
    \relative c' {
      cis'8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \\
```

```
    \relative c' {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative c' {
      <fis, a cis>8[ <fis a cis>
      \change Staff = up
      cis' cis
      \change Staff = down
      <fis, a> <fis a>]
      \showStaffSwitch
      \change Staff = up
      dis'4 |
      \change Staff = down
      <fis, a cis>4 gis <f a d>2 |
    }
  }
}

\new PianoStaff {
  <<
    \context Staff = "up" {
      \accidentalStyle default
      \musicA
    }
    \context Staff = "down" {
      \accidentalStyle default
      \musicB
    }
  >>
}
```



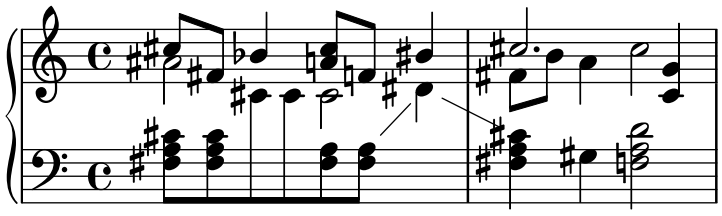
Si noti che le ultime linee di questo esempio possono essere sostituite dal seguente frammento, se si vuole usare lo stesso stile in entrambi i righi.

```
\new PianoStaff {
  <<
    \context Staff = "up" {
      %%% change the next line as desired:
      \accidentalStyle Score.default
```

```
        \musicA
      }
      \context Staff = "down" {
        \musicB
      }
    >>
  }
```

`default`

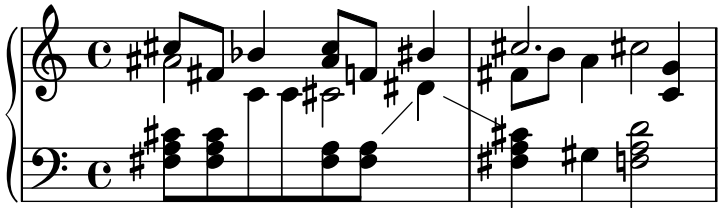
Questo è il comportamento predefinito del compositore tipografico. Corrisponde alla pratica comunemente impiegata dal diciottesimo secolo: le alterazioni vengono ricordate fino alla fine della misura in cui si trovano, limitatamente all'ottava di appartenenza. Quindi, nell'esempio seguente non compare alcun segno di bequadro prima del `b` nella seconda misura o prima dell'ultimo `c`:



`voice`

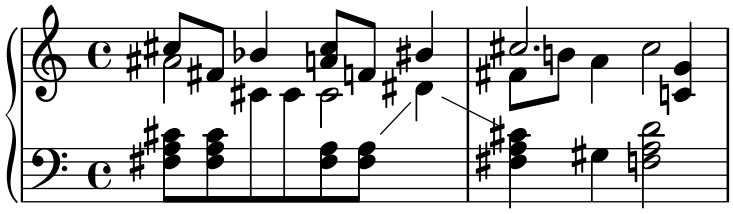
Normalmente le alterazioni mantengono la propria validità a livello di `Staff`. Tuttavia in questo stile le alterazioni vengono gestite individualmente per ogni voce. Al di fuori di quest'aspetto, lo stile è analogo a `default`.

Di conseguenza, le alterazioni relative a una voce non vengono cancellate nelle altre voci. Un risultato spesso non desiderabile: nell'esempio seguente è difficile capire se il secondo `a` sia naturale o diesis. L'opzione `voice` deve essere quindi usata solo se ogni voce è destinata a un esecutore diverso. Se la partitura deve essere letta da un unico musicista (come nel caso della partitura del direttore, o di uno spartito per pianoforte), allora è preferibile usare `modern` o `modern-cautionary`.



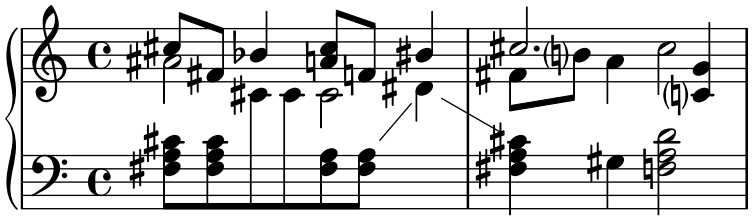
`modern`

Questa regola corrisponde alla pratica comune del ventesimo secolo. Omette i segni di bequadro supplementari che in passato erano di norma anteposti al diesis che segue un doppio diesis o a un bemolle che segue un doppio bemolle. La regola `modern` presenta le stesse alterazioni di `default`, con due aggiunte che servono a evitare ambiguità: i segni di annullamento delle alterazioni temporanee sono anteposti alle note sulla stessa ottava della misura successiva e alle note in ottave diverse nella stessa misura. In questo esempio, dunque, i bequadri del `b` e del `c` nella seconda misura del rigo superiore:



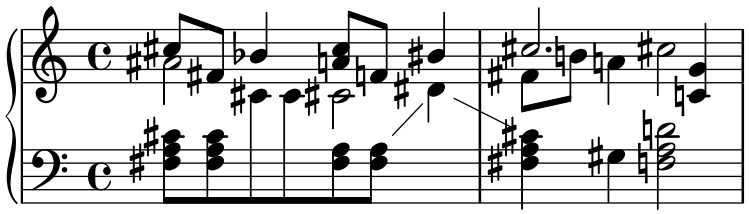
`modern-cautionary`

Questa regola è simile a `modern`, ma le alterazioni 'supplementari' (quelle non mostrate da `default`) sono segnate come alterazioni di precauzione. Di norma, sono poste tra parentesi; altrimenti, possono essere ridotte in corpo più piccolo definendo la proprietà `cautionary-style` di `AccidentalSuggestion`.



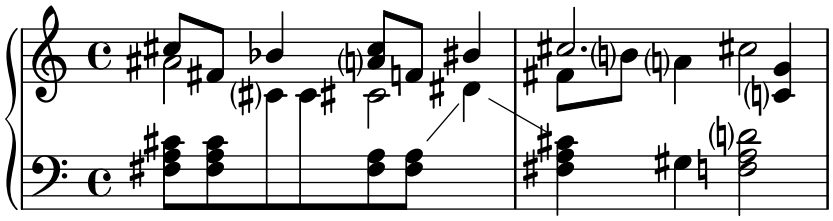
`modern-voice`

Questa regola viene usata per le alterazioni su più voci destinate sia agli esecutori che suonano una singola voce sia a quelli che suonano tutte le voci. Le alterazioni sono mostrate su tutte le voci, ma *sono annullate* su ogni voce dello stesso rigo (`Staff`). Quindi, l'alterazione dell' `a` nell'ultima misura viene annullata perché l'annullamento precedente si trovava in una voce diversa, mentre quella del `d` nel rigo inferiore viene annullata a causa dell'alterazione in un'altra voce della misura precedente:



`modern-voice-cautionary`

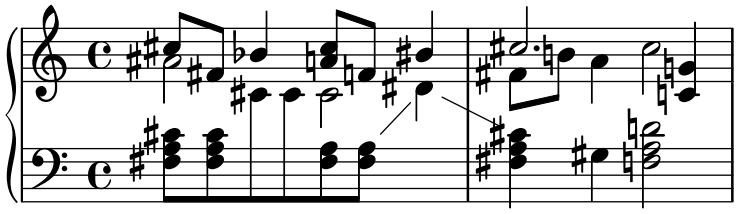
Questa regola è analoga a `modern-voice`, ma con le alterazioni supplementari (quelle non mostrate da `voice`) segnate come alterazioni di precauzione. Tutte le alterazioni mostrate da `default` *sono* mostrate con questa regola, ma alcune di esse sono indicate come alterazioni di precauzione.



`piano`

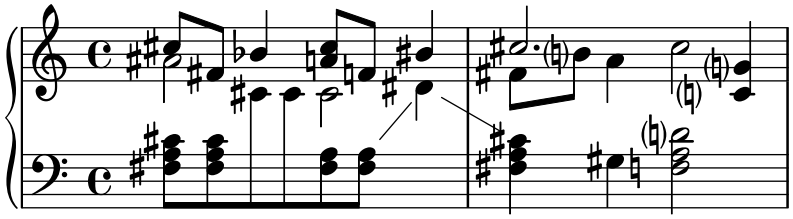
Questa regola riflette la pratica del ventesimo secolo per la notazione per pianoforte. Il suo comportamento è molto simile allo stile `modern`, ma in questo caso le alterazioni vengono annullate in tutti i righi che si trovano nello stesso `GrandStaff` o `PianoStaff`, dunque tutte gli annullamenti delle note finali.

È lo stile predefinito per gli attuali `GrandStaff` e `PianoStaff`.



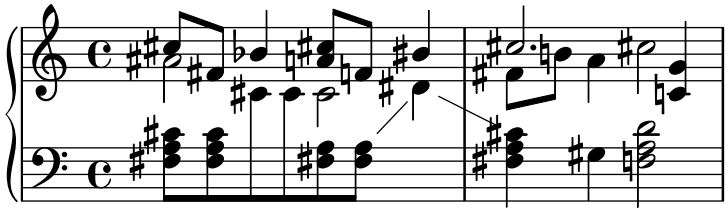
`piano-cautionary`

È uguale a `piano` ma con le alterazioni supplementari mostrate come alterazioni di precauzione.



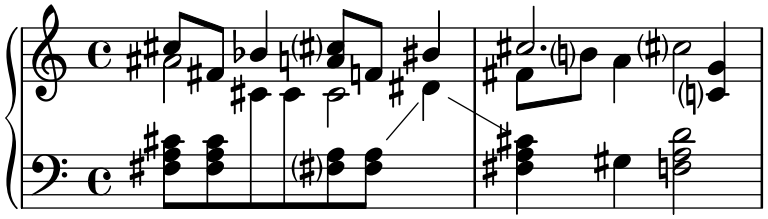
`neo-modern`

Questa regola si riferisce a una pratica tipica della musica contemporanea: le alterazioni sono mostrate come in `modern`, ma vengono ripetute se la stessa nota appare in seguito nella stessa misura – a meno che la seconda occorrenza non segua direttamente la prima.



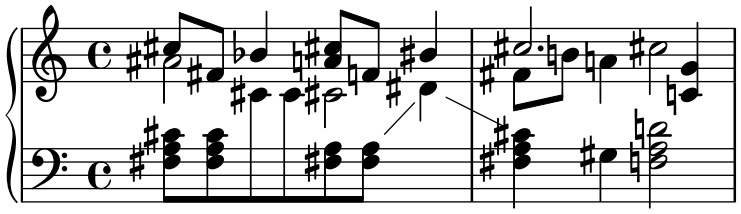
`neo-modern-cautionary`

Questa regola è simile a `neo-modern`, ma le alterazioni supplementari sono mostrate come alterazioni di precauzione.



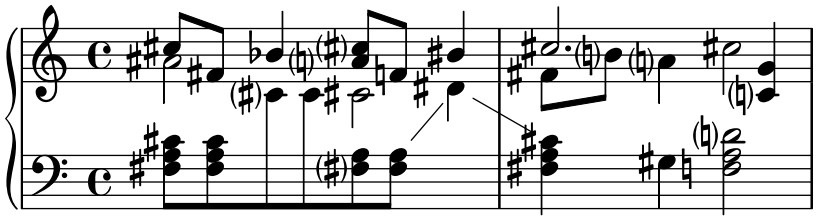
`neo-modern-voice`

Questa regola viene usata per le alterazioni su più di una voce che devono essere lette sia da musicisti che suonano una singola voce sia da musicisti che suonano tutte le voci. Le alterazioni per ogni voce sono mostrate come nello stile `neo-modern`, ma vengono annullate attraverso le voci nello stesso rigo (`Staff`).



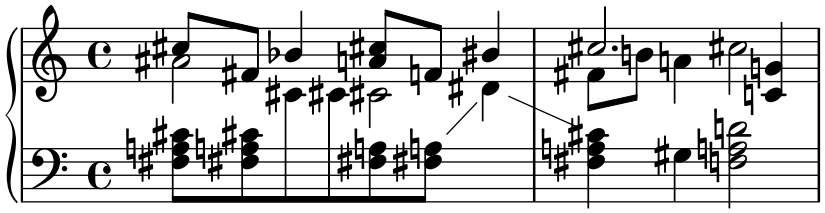
`neo-modern-voice-cautionary`

Questa regola è simile a `neo-modern-voice`, ma le alterazioni supplementari sono indicate come alterazioni di precauzione.



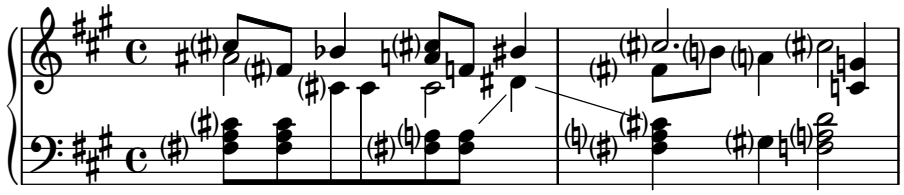
`dodecaphonic`

Questa regola riflette una regola introdotta dai compositori all'inizio del ventesimo secolo nel tentativo di abolire la gerarchia tra suoni naturali e non naturali. Con questo stile, *ogni* nota presenta un segno di alterazione, anche i suoni naturali.



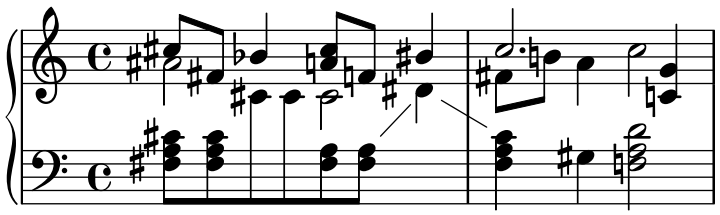
`teaching`

Questa regola è pensata per gli studenti: permette di generare facilmente degli spartiti di scale con le alterazioni di precauzione inserite in modo automatico. Alle alterazioni, indicate come nello stile `modern`, vengono aggiunte ulteriori segni di precauzione per tutti i diesis e bemolle specificati dall'armatura di chiave, fuorché nel caso di ripetizioni immediatamente successive di una stessa nota.



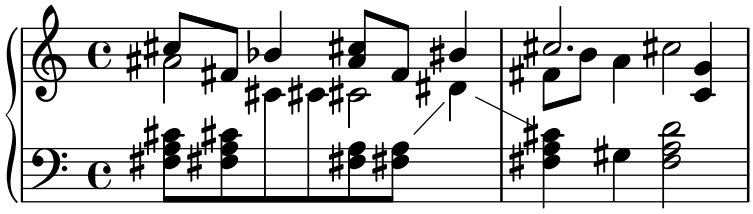
`no-reset`

È identico a `default`, ma le alterazioni mantengono la propria validità 'per sempre', non solo all'interno della singola misura:



`forget`

È il contrario di `no-reset`: le alterazioni non vengono ricordate affatto – pertanto, tutte le alterazioni si riferiscono all'armatura di chiave, indipendentemente dal materiale musicale precedente.



Vedi anche

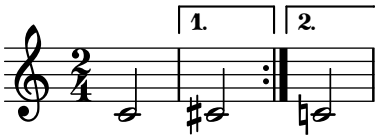
Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Accidental” in *Guida al Funzionamento Interno*, Sezione “Accidental_engraver” in *Guida al Funzionamento Interno*, Sezione “GrandStaff” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “AccidentalSuggestion” in *Guida al Funzionamento Interno*, Sezione “AccidentalPlacement” in *Guida al Funzionamento Interno*, Sezione “accidental-suggestion-interface” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

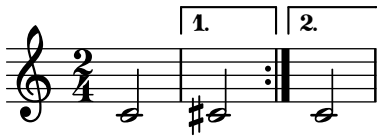
Le note simultanee non vengono considerate nell’individuazione automatica delle alterazioni; vengono prese come riferimento solo le note precedenti e l’armatura di chiave. Se la stessa nota occorre simultaneamente con alterazioni diverse, può essere necessario forzare le alterazioni con `!` o `?`: ‘`<f! fis!>`’.

L’annullamento di precauzione delle alterazioni avviene in relazione alla misura precedente. Tuttavia, nel blocco `\alternative` che segue una sezione `\repeat volta N`, è auspicabile che l’annullamento sia calcolato in base alla precedente misura *eseguita*, non alla precedente misura *stampata*. Nell’esempio seguente il Do naturale della seconda volta non richiede il segno di bequadro:



Si può usare il seguente espediente: si definisce una funzione che imposti localmente lo stile delle alterazioni su `forget`:

```
forget = #(define-music-function (parser location music) (ly:music?) #{
  \accidentalStyle forget
  #music
  \accidentalStyle modern
#})
{
  \accidentalStyle modern
  \time 2/4
  \repeat volta 2 {
    c'2
  }
  \alternative {
     cis'
     \forget c'
  }
}
```
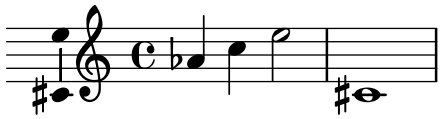


Ambitus

Il termine *ambitus* (pl. ambitus) indica l'ambito di altezze di una determinata voce all'interno di una composizione musicale. Può indicare anche l'estensione di uno strumento musicale, ovvero l'intera gamma di suoni che può produrre. L'ambitus viene usato nelle parti vocali in modo che gli esecutori possano capire facilmente se siano adeguate alle loro possibilità.

L'ambitus viene indicato all'inizio del brano, prima della chiave iniziale. L'intervallo è individuato graficamente da due teste di nota che rappresentano l'altezza più bassa e più alta. Le alterazioni sono mostrate solo se non fanno parte dell'armatura di chiave.

```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

\relative c'' {
  aes c e2
  cis,1
}
```
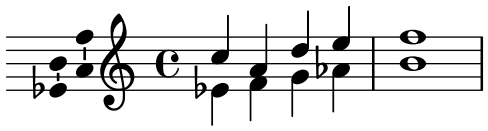


Frammenti di codice selezionati

Un ambitus per voce

L'ambitus può essere specificato per voce. In tal caso occorre spostarlo manualmente per evitare collisioni.
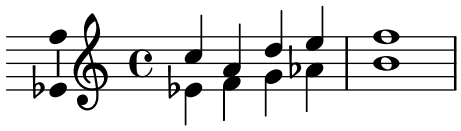
```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = #2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Ambitus su più voci

Se si aggiunge l'incisore `Ambitus_engraver` al contesto `Staff` viene creato un solo ambitus per il rigo, anche nel caso di righi che hanno più voci.

```
\new Staff \with {
  \consists "Ambitus_engraver"
  }
<<
  \new Voice \relative c'' {
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Modifica dell'intervallo dell'ambitus

È possibile cambiare le impostazioni predefinite dell'intervallo tra le teste di nota dell'ambitus e la linea che le collega.

```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

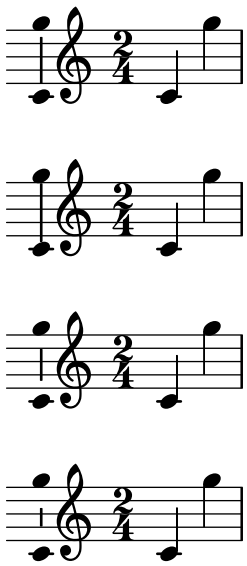
\new Staff {
  \time 2/4
  % Default setting
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #0
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1
  c'4 g''
```

```
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1.5
  c'4 g''
}
```



Vedi anche

Glossario musicale: Sezione “ambitus” in *Glossario Musicale*.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Ambitus_engraver” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “Ambitus” in *Guida al Funzionamento Interno*, Sezione “AmbitusAccidental” in *Guida al Funzionamento Interno*, Sezione “AmbitusLine” in *Guida al Funzionamento Interno*, Sezione “AmbitusNoteHead” in *Guida al Funzionamento Interno*, Sezione “ambitus-interface” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Le collisioni non vengono gestite in presenza di un ambitus multiplo su più di una voce.

1.1.4 Teste di nota
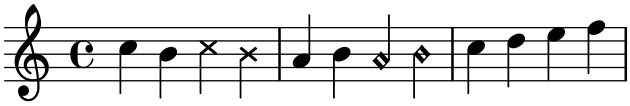
Questa sezione suggerisce i modi in cui modificare la testa di una nota.

Teste di nota speciali

L’aspetto delle teste delle note può essere modificato:

```
c4 b
\override NoteHead.style = #'cross
c4 b
\revert NoteHead.style
a b
\override NoteHead.style = #'harmonic
a b
```

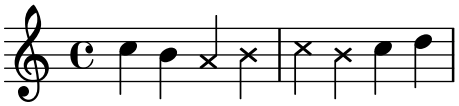
```
\revert NoteHead.style
c4 d e f
```



L'elenco di tutti gli stili per le teste di nota è in Sezione A.9 [Note head styles], pagina 653.

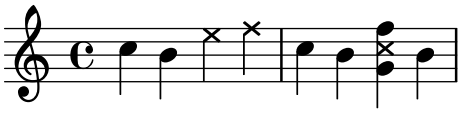
Lo stile barrato (`cross`) viene usato per rappresentare varie intenzioni musicali. I seguenti comandi generici predefiniti modificano la testa della nota nei contesti del rigo e dell'intavolatura e possono essere usati per rappresentare qualsiasi significato musicale:

```
c4 b
\xNotesOn
 a b c4 b
\xNotesOff
c4 d
```



Questo comando può essere usato all'interno e all'esterno degli accordi per generare teste barrate sia nel contesto del rigo che in quello dell'intavolatura:
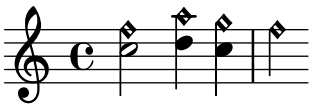
```
c4 b
\xNote { e f }
c b < g \xNote c f > b
```



Potete utilizzare, al posto di `\xNote`, `\xNotesOn` e `\xNotesOff`, i comandi `\deadNote`, `\deadNotesOn` e `\deadNotesOff`. Il termine *dead note* è di uso comune tra i chitarristi.

Esiste anche una scorciatoia simile per le forme a diamante:

```
<c f\harmonic>2 <d a'\harmonic>4 <c g'\harmonic> f\harmonic
```



Comandi predefiniti

`\harmonic`, `\xNotesOn`, `\xNotesOff`, `\xNote`.

Vedi anche

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

Guida alla notazione: Sezione A.9 [Note head styles], pagina 653, ⟨undefined⟩ [Chorded notes], pagina ⟨undefined⟩, [Indicating harmonics and dampened notes], pagina 366.

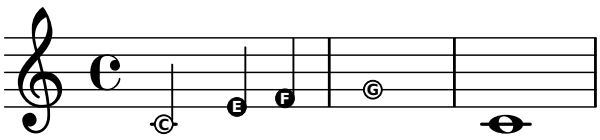
Guida al funzionamento interno: Sezione "note-event" in *Guida al Funzionamento Interno*, Sezione "Note_heads_engraver" in *Guida al Funzionamento Interno*, Sezione "Ledger_line_engraver" in *Guida al Funzionamento Interno*, Sezione "NoteHead" in *Guida al*

Funzionamento Interno, Sezione “LedgerLineSpanner” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*, Sezione “ledger-line-spanner-interface” in *Guida al Funzionamento Interno*.

Testa di nota con nome della nota

La nota ‘easy play’ inserisce il nome della nota dentro la testa. Viene usata nella musica per principianti. Per rendere le lettere leggibili, occorrerebbe usare un carattere più grande. A questo proposito si veda Sezione 4.2.2 [Setting the staff size], pagina 514.

```
#(set-global-staff-size 26)
\relative c' {
  \easyHeadsOn
  c2 e4 f
  g1
  \easyHeadsOff
  c,1
}
```



Comandi predefiniti

`\easyHeadsOn`, `\easyHeadsOff`.

Frammenti di codice selezionati

Numeri dentro le teste di nota

Le teste di nota con nome della nota usano la proprietà `note-names` dell’oggetto `NoteHead` per determinare cosa appaia all’interno della testa. È possibile sovrascrivere questa proprietà e mostrare numeri corrispondenti ai gradi della scala.

Si può creare un semplice incisore che faccia questo per ogni oggetto testa di nota che incontra.

```
#(define Ez_numbers_engraver
   (make-engraver
    (acknowledgers
     ((note-head-interface engraver grob source-engraver)
      (let* ((context (ly:translator-context engraver))
             (tonic-pitch (ly:context-property context 'tonic))
             (tonic-name (ly:pitch-notename tonic-pitch))
             (grob-pitch
              (ly:event-property (event-cause grob) 'pitch))
             (grob-name (ly:pitch-notename grob-pitch))
             (delta (modulo (- grob-name tonic-name) 7))
             (note-names
              (make-vector 7 (number->string (1+ delta)))))
        (ly:grob-set-property! grob 'note-names note-names))))))

#(set-global-staff-size 26)

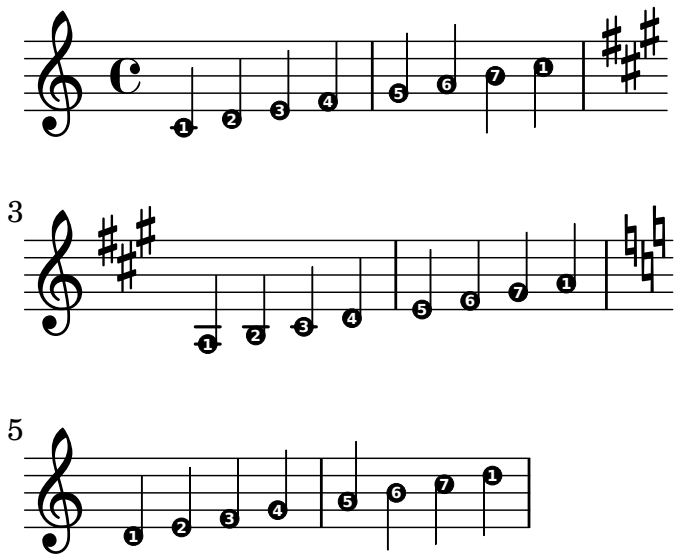
\layout {
  ragged-right = ##t
```

```
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

\relative c' {
  \easyHeadsOn
  c4 d e f
  g4 a b c \break

  \key a \major
  a,4 b cis d
  e4 fis gis a \break

  \key d \dorian
  d,4 e f g
  a4 b c d
}
```



Vedi anche

Guida alla notazione: Sezione 4.2.2 [Setting the staff size], pagina 514.
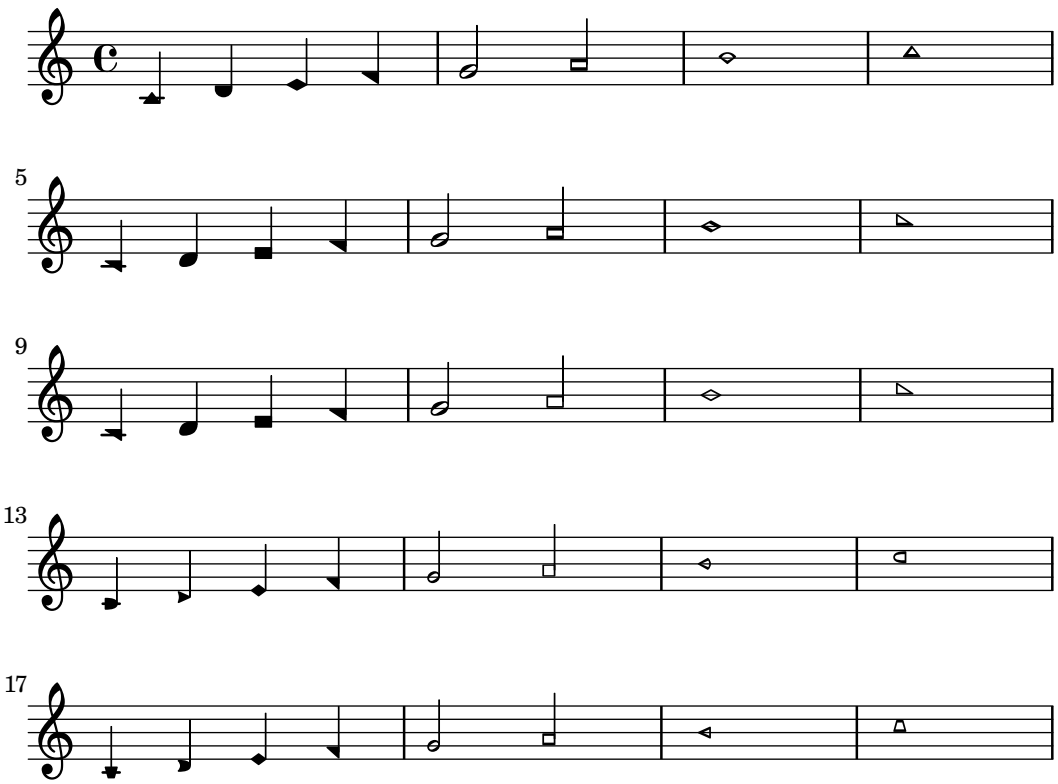
Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*.

Teste di nota a forma variabile
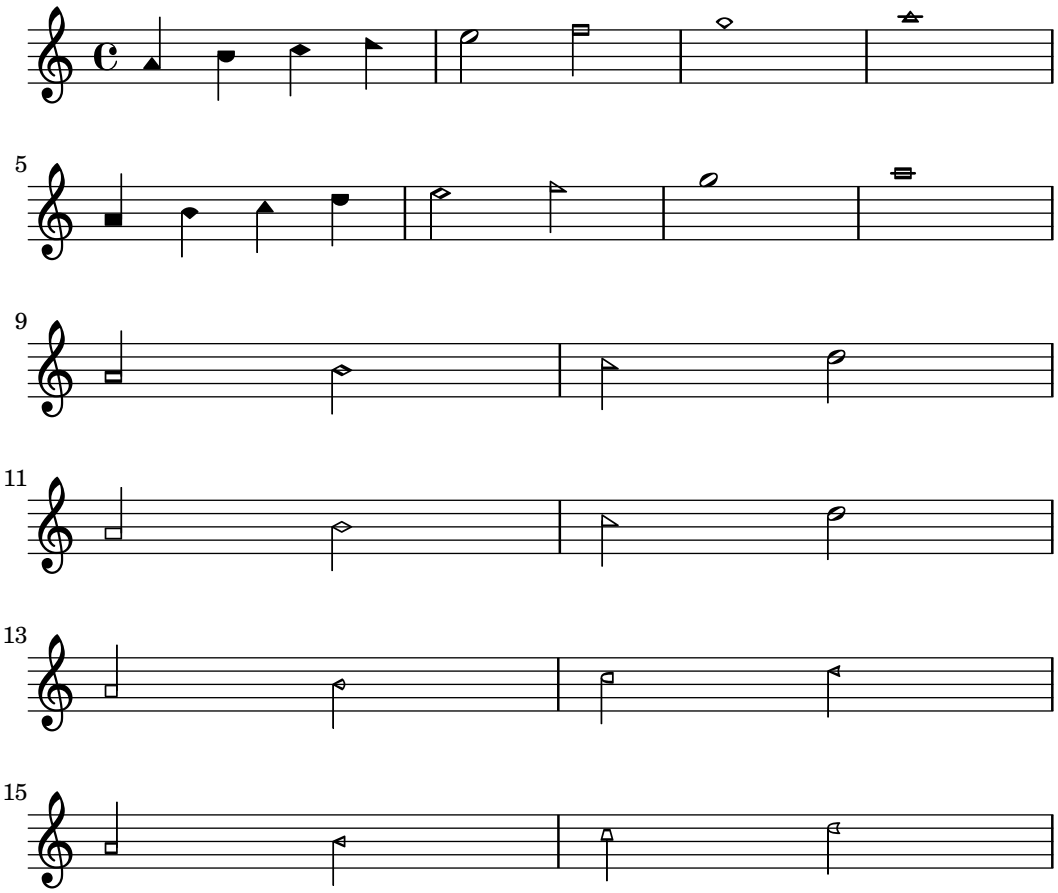
In alcune notazioni, la forma della testa della nota corrisponde alla funzione armonica di una nota nella scala. Questa notazione era comune nei canzonieri americani del diciannovesimo secolo. Gli stili possibili sono Sacred Harp, Southern Harmony, Funk (Harmonica Sacra), Walker e Aiken (Christian Harmony):

```
\aikenHeads
c, d e f g2 a b1 c \break
\sacredHarpHeads
c,4 d e f g2 a b1 c \break
\southernHarmonyHeads
c,4 d e f g2 a b1 c \break
\funkHeads
c,4 d e f g2 a b1 c \break
\walkerHeads
c,4 d e f g2 a b1 c \break
```



Le forme variano in base al grado della scala; la scala è determinata dal comando `\key`. Se si scrive in tonalità minore, il grado della scala può essere determinato in base alla relativa maggiore:

```
\key a \minor
\aikenHeads
a b c d e2 f g1 a \break
\aikenHeadsMinor
a,4 b c d e2 f g1 a \break
\sacredHarpHeadsMinor
a,2 b c d \break
\southernHarmonyHeadsMinor
a2 b c d \break
\funkHeadsMinor
a2 b c d \break
\walkerHeadsMinor
a2 b c d \break
```



Comandi predefiniti

`\aikenHeads`, `\aikenHeadsMinor`, `\funkHeads`, `\funkHeadsMinor`, `\sacredHarpHeads`, `\sacredHarpHeadsMinor`, `\southernHarmonyHeads`, `\southernHarmonyHeadsMinor`, `\walkerHeads`, `\walkerHeadsMinor`.

Frammenti di codice selezionati

Applicazione degli stili delle teste di nota in base al grado della scala

La proprietà `shapeNoteStyles` può essere usata per definire vari stili di teste di nota per ogni grado della scala (definita dall'armatura di chiave o dalla proprietà `tonic`). Questa proprietà richiede un insieme di simboli, che può essere puramente arbitrario (sono permesse espressioni geometriche come `triangle`, `cross` e `xcircle`) o basato sull'antica tradizione tipografica americana (sono consentiti anche alcuni nomi di nota latini).

Detto questo, per imitare gli antichi canzionieri americani, ci sono vari stili predefiniti disponibili attraverso dei comodi comandi come `\aikenHeads` o `\sacredHarpHeads`.

Questo esempio mostra modi diversi di ottenere teste di nota di varie forme e illustra la possibilità di trasporre una melodia senza perdere la corrispondenza tra le funzioni armoniche e gli stili delle teste.

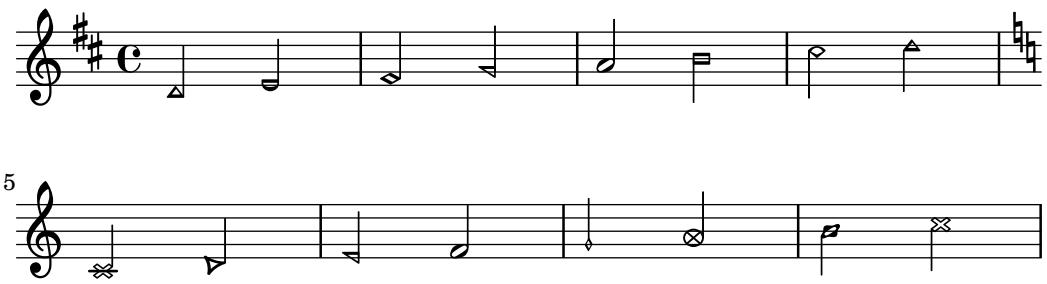
```
fragment = {
  \key c \major
  c2 d
  e2 f
  g2 a
  b2 c
}

\new Staff {
```

```
  \transpose c d
  \relative c' {
    \set shapeNoteStyles = ##(do re mi fa
                              #f la ti)
    \fragment
  }

  \break

  \relative c' {
    \set shapeNoteStyles = ##(cross triangle fa #f
                              mensural xcircle diamond)
    \fragment
  }
}
```



La lista completa di tutti gli stili delle teste si trova in Sezione A.9 [Note head styles], pagina 653.

Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida alla notazione: Sezione A.9 [Note head styles], pagina 653.

Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*.

Improvvisazione

L’improvvisazione viene talvolta indicata con teste tagliate: l’esecutore può scegliere qualsiasi nota ma deve seguire il ritmo indicato. Si possono creare queste teste:

```
\new Voice \with {
  \consists "Pitch_squash_engraver"
} {
  e8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  e2 ~ e8 f4 f8 ~
  f2
  \improvisationOff
  a16( bes) a8 g e
}
```



Comandi predefiniti
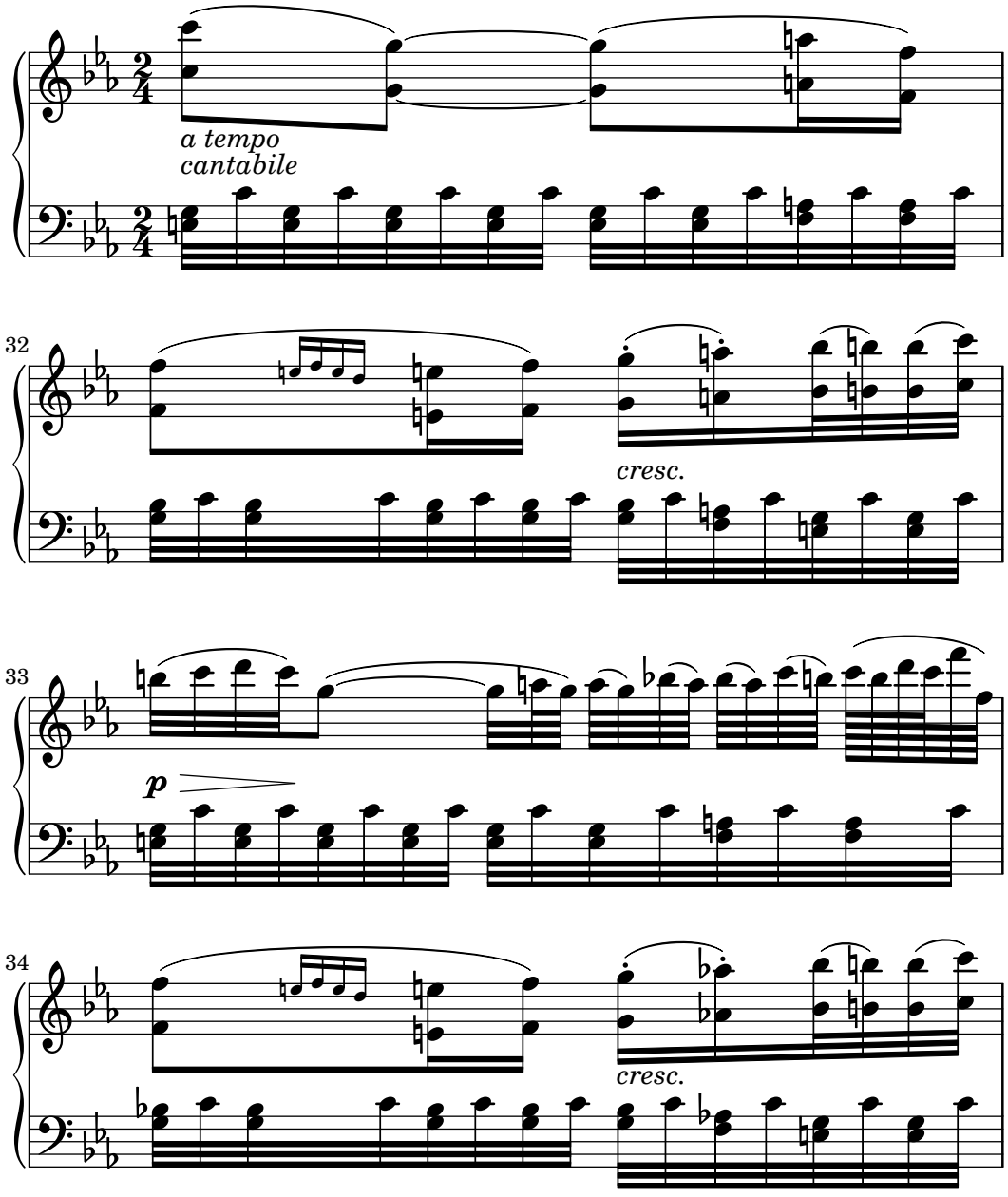
`\improvisationOn`, `\improvisationOff`.

Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Pitch_squash_engraver” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*, Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*.

1.2 Ritmi
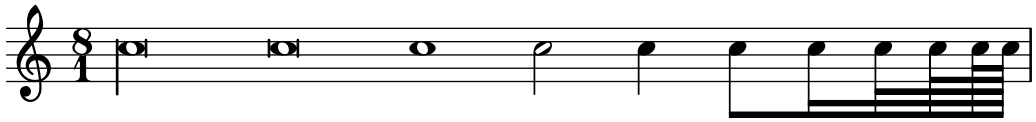


Questa sezione tratta i ritmi, le pause, le durate, la disposizione delle travature e le battute.

1.2.1 Inserimento delle durate

Durata
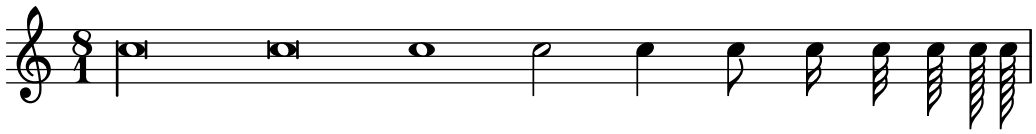
Le durate, indicate con numeri e punti, sono indicate con i valori corrispondenti. Per esempio, una nota di un quarto si indica con un `4` (dato che il suo valore è un 1/4), mentre una minima si indica col `2` (dato che il suo valore è 1/2). Per le note più lunghe di un intero bisogna usare i comandi `\longa` (due volte una breve) e `\breve`. La minor durata esprimibile per una nota indipendente è di 128; è possibile inserire anche valori inferiori, ma solo all'interno di travature.

```
\time 8/1
c\longa c\breve c1 c2
c4 c8 c16 c32 c64 c128 c128
```



Ecco gli stessi valori con la disposizione automatica delle travature disabilitata.
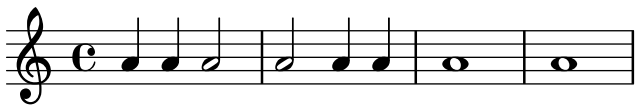
```
\time 8/1
\autoBeamOff
c\longa c\breve c1 c2
c4 c8 c16 c32 c64 c128 c128
```



Solo nella notazione per musica antica è possibile specificare una nota che dura quattro volte una breve, attraverso il comando `\maxima`. Per maggiori informazioni si veda Sezione 2.9 [Ancient notation], pagina 413.
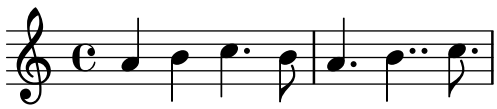
Se una durata viene omessa, viene mantenuta quella precedente. Il valore predefinito della prima nota è di un quarto.

```
a a a2 a a4 a a1 a
```



Per ottenere note puntate, si inserisce un punto (`.`) dopo la durata. Le figure con doppio punto si indicano aggiungendo due punti, e così via.

```
a4 b c4. b8 a4. b4.. c8.
```



Alcune durate non possono essere rappresentate soltanto con durate e punti; occorre inserire una legatura di valore tra due o più note. I dettagli sono spiegati in ⟨undefined⟩ [Ties], pagina ⟨undefined⟩.

Per sapere come specificare le durate delle sillabe del testo e come allineare il testo alle note, si veda Sezione 2.1 [Vocal music], pagina 247.

Le note possono essere distanziate in modo rigorosamente proporzionale alla loro durata. I dettagli relativi a questo argomento e alle impostazioni della notazione proporzionale si trovano in Sezione 4.5.5 [Proportional notation], pagina 543.

Di norma i punti sono spostati in su per evitare le linee del rigo, fuorché all'interno di passaggi polifonici. I punti possono essere orientati manualmente verso l'alto o verso il basso; si veda Sezione 5.4.2 [Direction and placement], pagina 585.

Comandi predefiniti

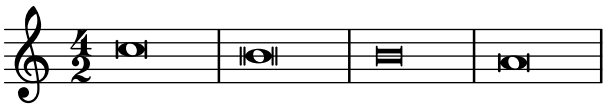
`\autoBeamOn`, `\autoBeamOff`, `\dotsUp`, `\dotsDown`, `\dotsNeutral`.

Frammenti di codice selezionati

Note brevi alternative

Le note brevi sono disponibili anche con due linee verticali su ciascun lato della testa invece di una sola e in stile barocco.
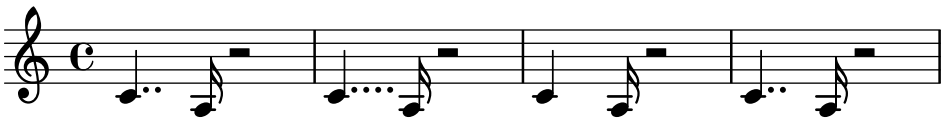
```
\relative c'' {
  \time 4/2
  c\breve |
  \override Staff.NoteHead.style = #'altdefault
  b\breve
  \override Staff.NoteHead.style = #'baroque
  b\breve
  \revert Staff.NoteHead.style
  a\breve
}
```



Modifica del numero di punti di aumentazione per nota

Il numero di punti di aumentazione su una singola nota può essere modificato in modo indipendente dai punti posizionati dopo la nota.

```
\relative c' {
  c4.. a16 r2 |
  \override Dots.dot-count = #4
  c4.. a16 r2 |
  \override Dots.dot-count = #0
  c4.. a16 r2 |
  \revert Dots.dot-count
  c4.. a16 r2 |
}
```



Vedi anche

Glossario musicale: Sezione "breve" in *Glossario Musicale*, Sezione "longa" in *Glossario Musicale*, Sezione "maxima" in *Glossario Musicale*, Sezione "durata" in *Glossario Musicale*, Sezione "Nomi di durata delle note e delle pause" in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Automatic beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Ties], pagina ⟨undefined⟩, ⟨undefined⟩ [Stems], pagina ⟨undefined⟩, ⟨undefined⟩ [Writing rhythms], pagina ⟨undefined⟩, ⟨undefined⟩ [Writing rests], pagina ⟨undefined⟩, Sezione 2.1 [Vocal music], pagina 247, Sezione 2.9 [Ancient notation], pagina 413, Sezione 4.5.5 [Proportional notation], pagina 543.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Dots" in *Guida al Funzionamento Interno*, Sezione "DotColumn" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Non c'è un limite massimo o minimo alla durata di una pausa, ma è il numero dei glifi ad essere limitato: si possono indicare pause da un centoventottesimo fino alla maxima (otto volte una semibreve).
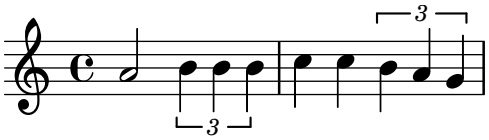
Gruppi irregolari

I gruppi irregolari sono costituiti da un'espressione musicale introdotta dal comando `\tuplet`, che moltiplica la velocità dell'espressione musicale per una frazione:

```
\tuplet frazione { musica }
```
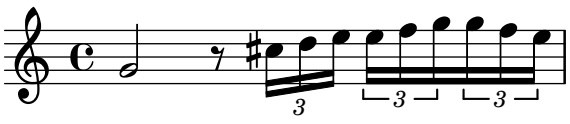
Il numeratore della frazione apparirà sopra o sotto le note; eventualmente, con l'aggiunta opzionale di una parentesi quadra. Il gruppo irregolare più comune è la terzina, in cui 3 note hanno la durata di 2:

```
a2 \tuplet 3/2 { b4 b b }
c4 c \tuplet 3/2 { b4 a g }
```



In caso di lunghi passaggi di gruppi irregolari, dover scrivere un comando `\tuplet` per ogni gruppo è scomodo. È possibile specificare direttamente la durata di un gruppo irregolare prima della musica per far sì che i gruppi siano suddivisi automaticamente:
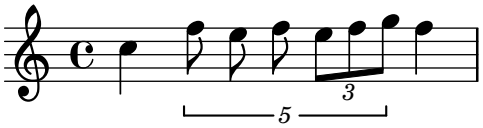
```
g2 r8 \tuplet 3/2 8 { cis16 d e e f g g f e }
```



Le parentesi dei gruppi irregolari si possono posizionare manualmente sopra o sotto il rigo, come spiegato dettagliatamente in Sezione 5.4.2 [Direction and placement], pagina 585.

È possibile annidare i gruppi irregolari:

```
\autoBeamOff
c4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[ f g] } } f4
```



La modifica di gruppi irregolari annidati che iniziano simultaneamente richiede l'uso di `\tweak`.

Per modificare la durata delle note senza introdurre un gruppo irregolare, si veda ⟨undefined⟩ [Scaling durations], pagina ⟨undefined⟩.

Comandi predefiniti

`\tupletUp`, `\tupletDown`, `\tupletNeutral`.

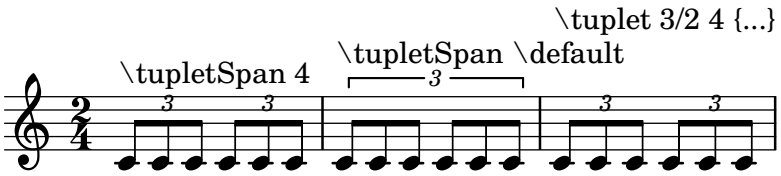
Frammenti di codice selezionati

Inserire vari gruppi irregolari usando una sola volta il comando `\tuplet`

La proprietà `tupletSpannerDuration` imposta la durata di ognuno dei gruppi irregolari compresi tra parentesi dopo il comando `\tuplet`. In questo modo si possono inserire molti gruppi irregolari consecutivi all'interno di una singola espressione `\tuplet`, risparmiando così tempo e spazio.

Ci sono vari modi per impostare `tupletSpannerDuration`. Il comando `\tupletSpan` la imposta su una certa durata e poi la annulla quando invece di una durata viene specificato `\default`. Altrimenti si può usare un argomento opzionale con `\tuplet`.

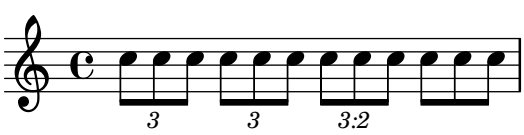
```
\relative c' {
  \time 2/4
  \tupletSpan 4
  \tuplet 3/2 { c8^"\\tupletSpan 4" c c c c c }
  \tupletSpan \default
  \tuplet 3/2 { c8^"\\tupletSpan \\default" c c c c c }
  \tuplet 3/2 4 { c8^"\\tuplet 3/2 4 {...}" c c c c c }
}
```



Modifica del numero del gruppo irregolare

Di norma compare sulla parentesi del gruppo irregolare solo il numeratore del numero del gruppo irregolare. Ma è possibile mostrare la frazione num:den del numero del gruppo irregolare oppure nascondere del tutto il numero.

```
\relative c'' {
  \tuplet 3/2 { c8 c c }
  \tuplet 3/2 { c8 c c }
  \override TupletNumber.text = #tuplet-number::calc-fraction-text
  \tuplet 3/2 { c8 c c }
  \omit TupletNumber
  \tuplet 3/2 { c8 c c }
}
```



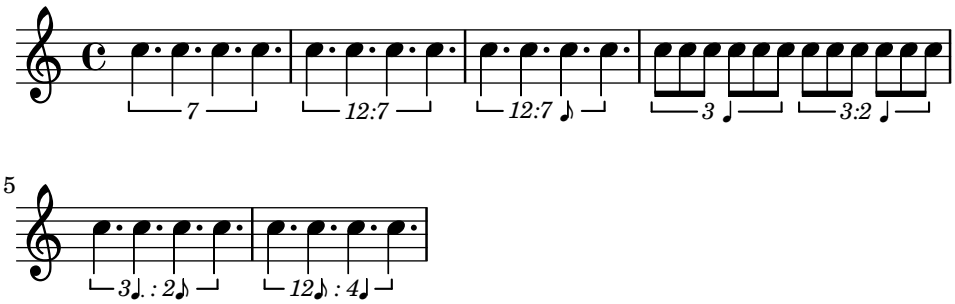
Numeri non predefiniti per i gruppi irregolari

LilyPond fornisce anche funzioni di formattazione che permettono di creare numeri di gruppi irregolari diversi dalla frazione vera e propria, così come di aggiungere un valore di nota al numero o alla frazione di un gruppo irregolare.

```
\relative c'' {
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7) "8")
  \tuplet 3/2  { c4. c4. c4. c4. }

  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text "4")
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text "4")
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }

  \once \override TupletNumber.text =
    #(tuplet-number::fraction-with-notes "4." "8")
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-fraction-with-notes 12 "8" 4 "4")
  \tuplet 3/2  { c4. c4. c4. c4. }
}
```



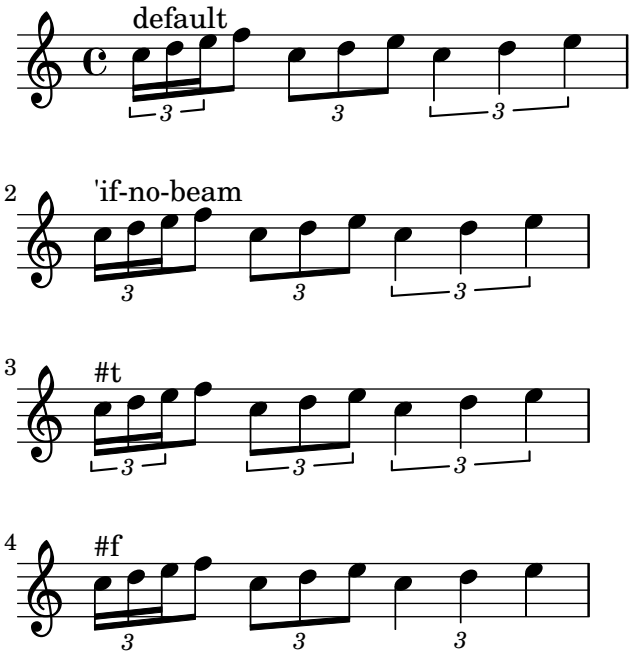
Controllare la visibilità della parentesi del gruppo irregolare

Il comportamento predefinito relativo alla visibilità della parentesi quadra del gruppo irregolare è di mostrare una parentesi a meno che non ci sia una travatura della stessa lunghezza del gruppo. Per controllare la visibilità di tale parentesi, si imposta la proprietà `'bracket-visibility` su `#t` (mostra sempre la parentesi), `#f` (non mostrare mai la parentesi) o `#'if-no-beam` (mostra la parentesi solo se non c'è una travatura).

```
music = \relative c'' {
  \tuplet 3/2 { c16[ d e } f8]
  \tuplet 3/2 { c8 d e }
  \tuplet 3/2 { c4 d e }
}

\new Voice {
```

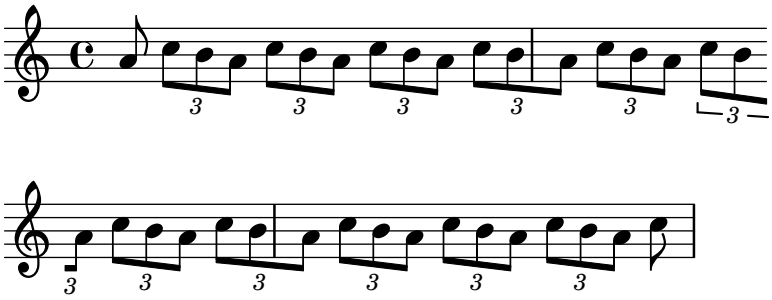
```
  \relative c' {
    << \music s4^"default" >>
    \override TupletBracket.bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket.bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket.bracket-visibility = ##f
    << \music s4^"#f" >>
  }
}
```



Consentire l'interruzione del rigo all'interno di gruppi irregolari con travature

Questo esempio artificioso mostra come permettere interruzioni del rigo sia manuali che automatiche all'interno di un gruppo irregolare con travature. Si noti che le travature di questi gruppi irregolari fuori dal ritmo devono essere disposte manualmente.

```
\layout {
  \context {
    \Voice
    % Permit line breaks within tuplets
    \remove "Forbid_line_break_engraver"
    % Allow beams to be broken at line breaks
    \override Beam.breakable = ##t
  }
}
\relative c'' {
  a8
  \repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
  % Insert a manual line break within a tuplet
  \tuplet 3/2 { c[ b \bar "" \break a] }
  \repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
  c8
}
```



Vedi anche

Glossario musicale: Sezione “terzina” in *Glossario Musicale*, Sezione “gruppo irregolare” in *Glossario Musicale*, Sezione “polimetrico” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Tweaking methods” in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Time administration], pagina ⟨undefined⟩, ⟨undefined⟩ [Scaling durations], pagina ⟨undefined⟩, Sezione 5.3.4 [The tweak command], pagina 579, ⟨undefined⟩ [Polymetric notation], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TupletBracket” in *Guida al Funzionamento Interno*, Sezione “TupletNumber” in *Guida al Funzionamento Interno*, Sezione “TimeScaledMusic” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Gli abbellimenti possono essere inseriti all’interno delle parentesi dei gruppi irregolari, *eccetto* quando un rigo inizia con un abbellimento seguito da un gruppo irregolare. In questo caso particolare, l’abbellimento deve essere inserito prima del comando `\tuplet` per evitare errori.
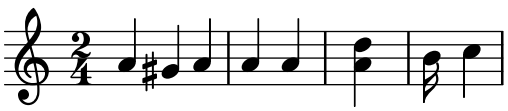
Quando si pone un gruppo irregolare all’inizio di un brano che presenta un’indicazione di `\tempo`, la musica deve essere inserita esplicitamente in un blocco `\new Voice`, come è spiegato in Sezione “Voices contain music” in *Manuale di Apprendimento*.

Scalare le durate

La durata di singole note, pause o accordi può essere moltiplicata per una frazione `N/M` aggiungendo `*N/M` (o `*N` se `M` è 1). Questo non cambierà l’aspetto delle note o delle pause, ma la durata così alterata verrà utilizzata per calcolare la posizione all’interno della misura e per impostare la durata nel file MIDI. Si possono combinare molteplici fattori, come `*L*M/N`. I fattori fanno parte della durata: quindi se non si specifica una durata per le note successive, la durata ripresa dalla nota precedente includerà il fattore di scalatura.

Nell’esempio seguente le prime tre note occupano esattamente due tempi, ma non sono indicate come gruppo irregolare.
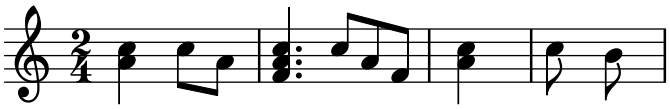
```
\time 2/4
% Trasforma le durate in terzine
a4*2/3 gis a
% Durate normali
a4 a
% Raddoppia la durata dell'accordo
<a d>4*2
% Durata di un quarto, ma appare come un sedicesimo
b16*4 c4
```



Anche la durata delle pause spaziatrici può essere modificata con un moltiplicatore. Può essere utile per saltare molte misure; per esempio `s1*23`.

Frammenti musicali più lunghi possono essere compressi secondo la stessa proporzione, come moltiplicando ogni nota, accordo o pausa per una medesima frazione. In questo modo, l'aspetto della musica non cambia ma la durata interna delle note viene moltiplicata per la frazione *num*/*den*. Ecco un esempio che mostra come la musica possa essere compressa e espansa:

```
\time 2/4
% Durate normali
<c a>4 c8 a
% Scala la musica di *2/3
\scaleDurations 2/3 {
  <c a f>4. c8 a f
}
% Scala la musica di *2
\scaleDurations 2/1 {
  <c' a>4 c8 b
}
```



Questo comando torna utile nella notazione polimetrica, si veda ⟨undefined⟩ [Polymetric notation], pagina ⟨undefined⟩.

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Tuplets], pagina ⟨undefined⟩, ⟨undefined⟩ [Invisible rests], pagina ⟨undefined⟩, ⟨undefined⟩ [Polymetric notation], pagina ⟨undefined⟩.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Problemi noti e avvertimenti

Il calcolo della posizione in una misura deve considerare tutti i fattori di dimensionamento applicati alle note di quella misura e gli esigui residui delle misure precedenti. Questo calcolo viene fatto con numeri razionali. Se un numeratore o un denominatore intermedi in quel calcolo eccedono di 2^30, l'esecuzione e la composizione tipografica si arresteranno in quel punto senza indicare un errore.
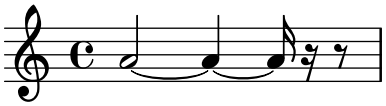
Legature di valore

Una legatura di valore connette le teste di due note della stessa altezza successive. Dunque, la legatura di valore prolunga la durata di una nota.

> **Nota:** Le legature di valore non devono essere confuse con le *legature di portamento*, che articolano un passaggio, o con le *legature di frase*, che delimitano una frase musicale. Una legatura di valore serve semplicemente a prolungare la durata di una nota, in modo analogo al punto di valore.
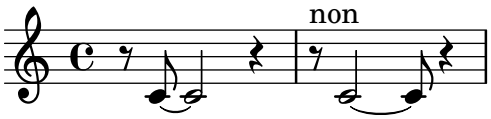
La legatura di valore si inserisce aggiungendo il simbolo tilde (`~`) alla prima di ogni coppia di note legate. Esso indica che la nota deve essere legata alla nota successiva, che deve essere della stessa altezza.

```
a2~ a4~ a16 r r8
```



Le legature di valore si usano per unire due note a cavallo di una stanghetta di battuta, oppure quando non si possono usare i punti per esprimere una particolare durata. Le legature si dovrebbero usare anche per unire note dalle durate superiori all'unità di suddivisione della misura:

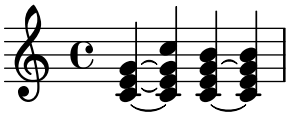
```
\relative c' {
  r8 c~ c2 r4 |
  r8^"non" c2~ c8 r4
}
```



Per legare una successione di note la cui durata si prolunga per più misure intere, è più semplice ricorrere alla suddivisione automatica delle note, come è spiegato in ⟨undefined⟩ [Automatic note splitting], pagina ⟨undefined⟩. Questo metodo divide automaticamente le note lunghe e le connette da misura a misura.

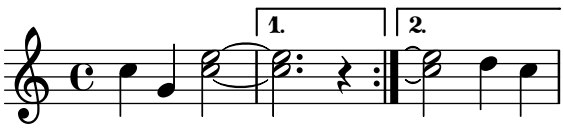
Quando si applica una legatura di valore a degli accordi, vengono legate tutte le teste delle note della stessa altezza. In assenza di altezze corrispondenti, non verrà creata alcuna legatura. Singoli suoni degli accordi possono essere legati inserendo la legatura all'interno dell'accordo stesso.

```
<c e g>~ <c e g c>
<c~ e g~ b> <c e g b>
```



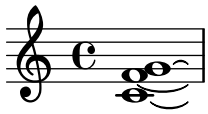
Quando la battuta della "seconda volta" di un ritornello inizia con una nota legata a quella precedente, occorre indicare la legatura nel modo seguente:

```
\repeat volta 2 { c g <c e>2~ }
\alternative {
  % Prima volta: la nota seguente viene legata in modo normale
  { <c e>2. r4 }
  % Seconda volta: la nota seguente ha una legatura ripetuta
  { <c e>2\repeatTie d4 c } }
```



Le legature *L.v.* (*laissez vibrer*) indicano che le note non devono essere terminate nettamente. Si usa nella notazione per pianoforte, arpa e altri strumenti a corda e a percussione. Si inseriscono così:
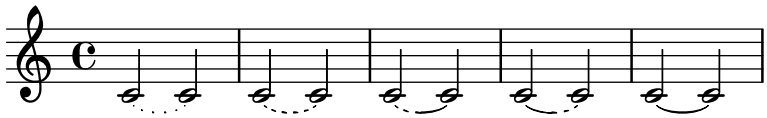
```
<c f g>1\laissezVibrer
```



Le legature di valore possono essere impostate manualmente per avere la curva in su o in giù, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 585.
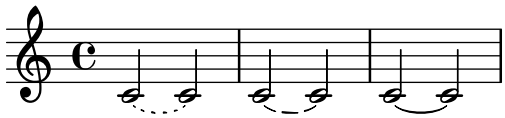
Le legature di valore possono essere tratteggiate, punteggiate, oppure tracciate secondo una successione di tratti continui e tratti interrotti.

```
\tieDotted
c2~ c
\tieDashed
c2~ c
\tieHalfDashed
c2~ c
\tieHalfSolid
c2~ c
\tieSolid
c2~ c
```



Si possono specificare modelli di tratteggiatura personalizzati:

```
\tieDashPattern #0.3 #0.75
c2~ c
\tieDashPattern #0.7 #1.5
c2~ c
\tieSolid
c2~ c
```
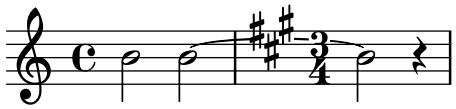


Le definizioni dei modelli di tratteggiatura delle legature di valore hanno la stessa struttura di quelle per le legature di portamento. I dettagli relativi ai modelli complessi di tratteggiatura sono trattati in ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩.

Se le legature collidono con altri oggetti del rigo, si possono sovrascrivere le proprietà di formattazione *whiteout* e *layer*.

```
\override Tie.layer = #-2
\override Staff.TimeSignature.layer = #-1
\override Staff.KeySignature.layer = #-1
\override Staff.TimeSignature.whiteout = ##t
\override Staff.KeySignature.whiteout = ##t
b2 b~
\time 3/4
\key a \major
```

```
b r4
```



Comandi predefiniti
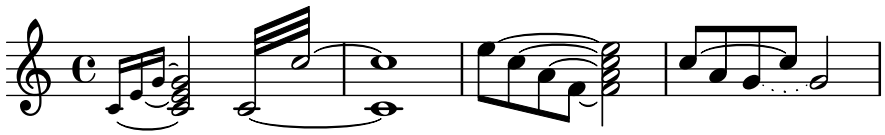
`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieDashPattern`, `\tieHalfDashed`, `\tieHalfSolid`, `\tieSolid`.

Frammenti di codice selezionati

Usare le legature di valore con un arpeggio

Le legature di valore vengono usate talvolta per scrivere un arpeggio. In questo caso, le due note da legare devono non essere consecutive. Per ottenere tale risultato occorre impostare la proprietà `tieWaitForNote` su `#t`. Questa funzionalità serve anche a legare un tremolo a un accordo e in generale qualsiasi coppia di note consecutive.

```
\relative c' {
  \set tieWaitForNote = ##t
  \grace { c16[ ~ e ~ g] ~ } <c, e g>2
  \repeat tremolo 8 { c32 ~ c' ~ } <c c,>1
  e8 ~ c ~ a ~ f ~ <e' c a f>2
  \tieUp
  c8 ~ a
  \tieDown
  \tieDotted
  g8 ~ c g2
}
```



Disegnare manualmente le legature di valore

Le legature di valore possono essere disegnate a mano cambiando la proprietà `tie-configuration` dell'oggetto `TieColumn`. Il primo numero indica la distanza dal centro del rigo nell'unità di metà spazio rigo, mentre il secondo numero indica la direzione (1 = su, -1 = giù).

```
\relative c' {
  <c e g>2~ <c e g>
  \override TieColumn.tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>2~ <c e g>
}
```



Vedi anche

Glossario musicale: Sezione “legatura di valore” in *Glossario Musicale*, Sezione “laissez vibrer” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic note splitting], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Expressive marks” in *Frammenti di codice*, Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “LaissezVibrerTie” in *Guida al Funzionamento Interno*, Sezione “LaissezVibrerTieColumn” in *Guida al Funzionamento Interno*, Sezione “TieColumn” in *Guida al Funzionamento Interno*, Sezione “Tie” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Cambiare rigo mentre una legatura di valore è attiva non produce una legatura obliqua.

Il cambio di chiave o di ottava durante una legatura di valore non è una situazione ben definita. In questi casi è preferibile usare una legatura di portamento.

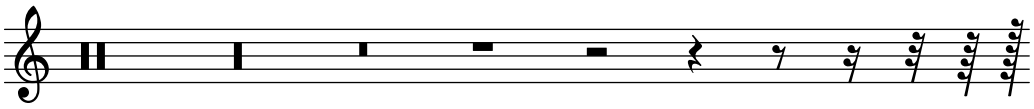
1.2.2 Inserimento delle pause

Le pause si inseriscono insieme alla musica contenuta nelle espressioni musicali.

Pause

Le pause si inseriscono allo stesso modo delle note, ma con il carattere `r`. Le durate più lunghe di un intero usano i seguenti comandi predefiniti:
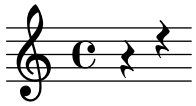
```
\new Staff {
  % Queste due linee servono solo ad abbellire questo esempio
  \time 16/1
  \omit Staff.TimeSignature
  % Mostra una pausa di maxima, equivalente a quattro brevi
  r\maxima
  % Mostra una pausa di longa, equivalente a due brevi
  r\longa
  % Mostra una pausa di breve
  r\breve
  r1 r2 r4 r8 r16 r32 r64 r128
}
```



Le pause d’intero, poste al centro della misura, devono essere inserite come pause multiple. Si possono usare sia per una sola misura sia su più misure, come è spiegato in ⟨undefined⟩ [Full measure rests], pagina ⟨undefined⟩.

Per indicare esplicitamente la posizione verticale di una pausa, si scrive la nota corrispondente seguita da `\rest`. Una pausa della durata della nota verrà collocata nella posizione della nota sul rigo. Questo permette una precisa formattazione manuale della musica polifonica, dato che il formattatore automatico che gestisce le collisioni tra pause non interviene su questo tipo di pause.

```
a4\rest d4\rest
```



Frammenti di codice selezionati

Stili di pausa

Esistono vari stili di pausa.

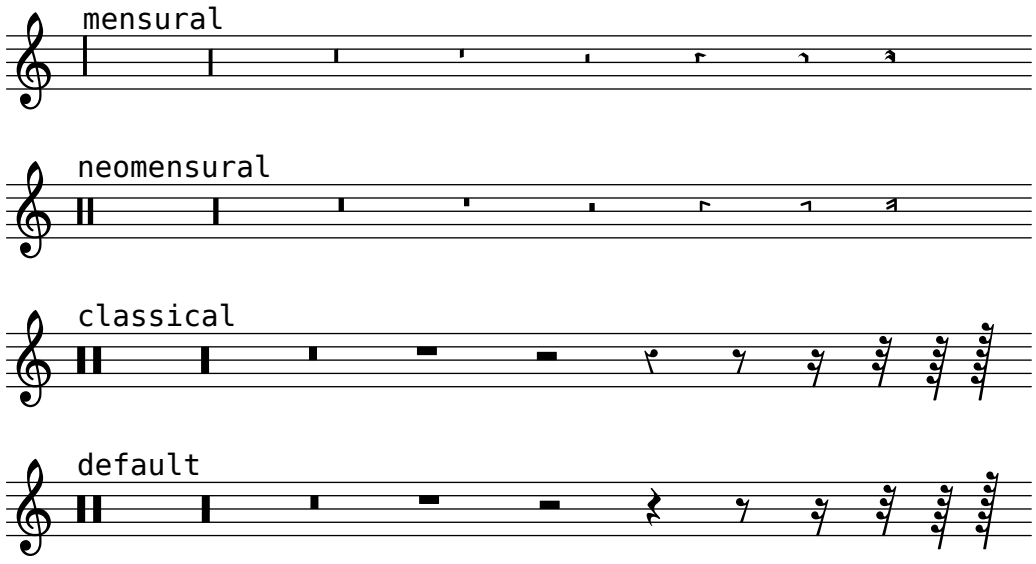
```
\layout {
  indent = 0
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}

\new Staff \relative c {
  \cadenzaOn
  \override Staff.Rest.style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""

  \override Staff.Rest.style = #'neomensural
  r\maxima^\markup \typewriter { neomensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""

  \override Staff.Rest.style = #'classical
  r\maxima^\markup \typewriter { classical }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
  \bar ""

  \override Staff.Rest.style = #'default
  r\maxima^\markup \typewriter { default }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
}
```



Vedi anche

Glossario musicale: Sezione “breve” in *Glossario Musicale*, Sezione “longa” in *Glossario Musicale*, Sezione “maxima” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Full measure rests], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Rest” in *Guida al Funzionamento Interno*.
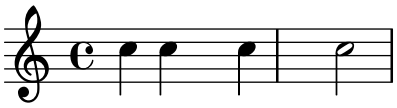
Problemi noti e avvertimenti

Non c’è un limite massimo o minimo alla durata di una pausa, ma è il numero dei glifi ad essere limitato: si possono indicare pause da un centoventottesimo fino alla maxima (otto volte una semibreve).

Pause invisibili
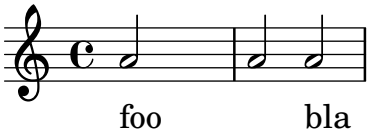
Una pausa invisibile (chiamata anche ‘pausa spaziatrice’) si inserisce come come una nota col nome `s`:

```
c4 c s c
s2 c
```



Le pause spaziatrici possono essere usate soltanto nella modalità note e nella modalità accordi. In altre situazioni, ad esempio quando si inserisce il testo vocale, si usa il comando `\skip` per saltare un valore musicale. `\skip` richiede una durata esplicita, ma questo requisito viene ignorato se il testo desume le proprie durate dalle note presenti in una melodia ad esso associata attraverso `\addlyrics` o `\lyricsto`.

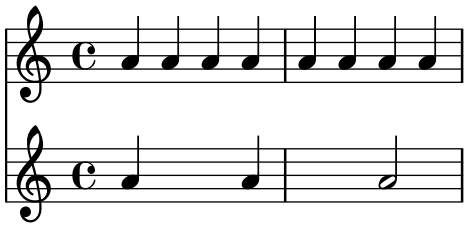
```
<<
  {
    a2 \skip2 a2 a2
  }
  \new Lyrics {
    \lyricmode {
      foo2 \skip 1 bla2
    }
  }
>>
```



Dato che `\skip` è un comando, non modifica la durata predefinita delle note che seguono, diversamente da `s`.

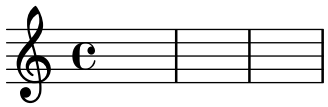
```
<<
  {
    \repeat unfold 8 { a4 }
  }
  {
    a4 \skip 2 a |
```

```
    s2 a
  }
>>
```



Una pausa spaziatrice crea implicitamente i contesti `Staff` e `Voice` se non esistono già, proprio come accade per le note e le pause:

```
s1 s s
```



`\skip` si limita a saltare un valore musicale, non crea nessun tipo di output.

```
% Questo input è corretto, ma non produce niente
\skip 1 \skip1 \skip 1
```

Vedi anche

Manuale di apprendimento: Sezione “Visibility and color of objects” in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Hidden notes], pagina ⟨undefined⟩, Sezione 5.4.6 [Visibility of objects], pagina 592.

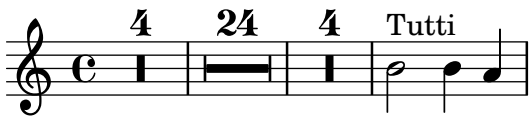
Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SkipMusic” in *Guida al Funzionamento Interno*.

Pause d’intero
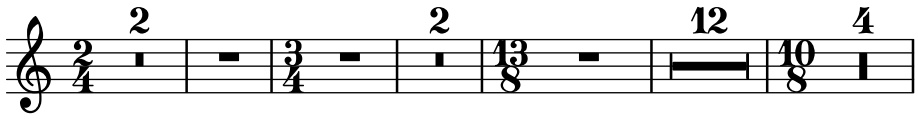
Le pause per una o più misure d’intero si inseriscono, come le note, col carattere maiuscolo `R`:

```
% L'insieme delle misure di pausa vengono riportate in una sola misura
\compressFullBarRests
R1*4
R1*24
R1*4
b2^"Tutti" b4 a4
```
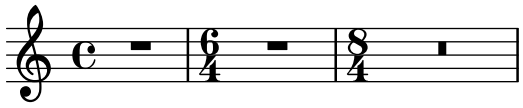


La durata delle pause multiple è identica alla notazione di durata usata per le note e deve essere sempre un numero intero di misure/lunghezze, quindi occorre spesso usare dei punti di aumentazione o delle frazioni:

```
\compressFullBarRests
\time 2/4
R1 | R2 |
\time 3/4
R2. | R2.*2 |
\time 13/8
R1*13/8 | R1*13/8*12 |
\time 10/8
R4*5*4 |
```



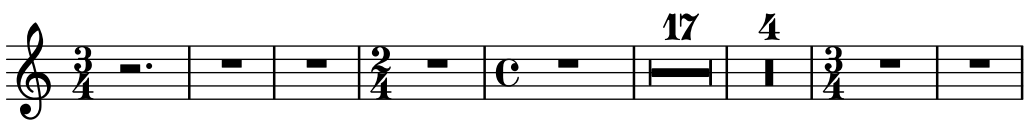
Una pausa d'intero appare al centro della misura con la durata di una semibreve o di una breve, in base all'indicazione di tempo.

```
\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |
```



Di norma una pausa multipla viene scorporata sul pentagramma in modo da mostrare esplicitamente tutte le misure per cui si prolunga. Altrimenti, è possibile indicarla collocando in una sola misura un simbolo di pausa multipla, col numero di misure per cui la pausa si prolunga posto al di sopra della misura stessa:

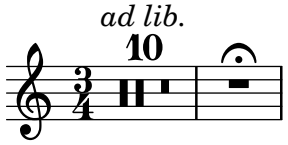
```
% Default behavior
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Tutte le misure di pausa sono riportate in una singola misura
\compressFullBarRests
r1 | R1*17 | R1*4 |
% Le misure della pausa multipla sono scorporate
\expandFullBarRests
\time 3/4
R2.*2 |
```



Si possono aggiungere delle annotazioni alle pause multiple. Il comando predefinito `\fermataMarkup` permette di aggiungere il segno di corona.

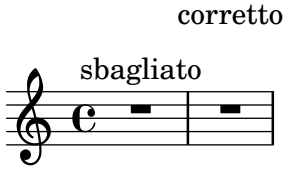
```
\compressFullBarRests
\time 3/4
```

```
R2.*10^\markup { \italic "ad lib." }
R2.^\fermataMarkup
```



Nota: Il testo connesso a una pausa multipla è un oggetto di tipo `MultiMeasureRestText`, non `TextScript`. Le sovrascritture devono specificare l'oggetto corretto o saranno ignorate. Si veda l'esempio seguente:

```
% Questo non funziona, perché è specificato il nome dell'oggetto sbagliato
\override TextScript.padding = #5
R1^"sbagliato"
% Questo è il nome dell'oggetto corretto da specificare
\override MultiMeasureRestText.padding = #5
R1^"corretto"
```



Quando una pausa multipla segue immediatamente un comando `\partial`, potrebbero non apparire i relativi avvertimenti del controllo battuta.

Comandi predefiniti

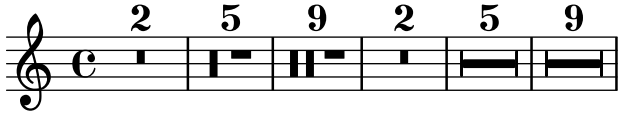
`\textLengthOn`, `\textLengthOff`, `\fermataMarkup`, `\compressFullBarRests`, `\expandFullBarRests`.

Frammenti di codice selezionati

Modificare la forma delle pause multiple

Se la pausa multipla dura dieci misure o un numero inferiore a dieci, nel rigo apparirà una serie di pause di longa e di breve (chiamate in tedesco "Kirchenpausen" - pause ecclesiastiche); altrimenti apparirà una semplice linea. Il numero predefinito di dieci può essere cambiato sovrascrivendo la proprietà `expand-limit`..

```
\relative c'' {
  \compressFullBarRests
  R1*2 | R1*5 | R1*9
  \override MultiMeasureRest.expand-limit = #3
  R1*2 | R1*5 | R1*9
}
```



Posizionamento delle pause multiple
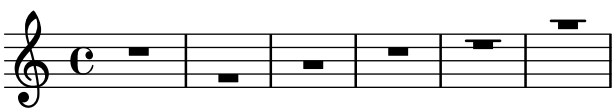
Diversamente dalle pause normali, non esiste un comando predefinito per cambiare la posizione sul rigo di un simbolo di pausa multipla di qualsiasi tipo connettendolo a una nota. Tuttavia, nella musica polifonica le pause multiple nelle voci dispari e pari sono separate verticalmente. Il posizionamento delle pause multiple si controlla nel modo seguente:

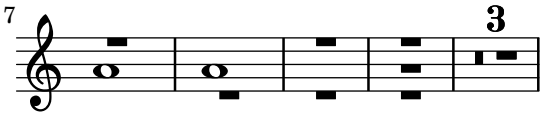
```
\relative c'' {
  % Multi-measure rests by default are set under the fourth line
  R1
  % They can be moved using an override
  \override MultiMeasureRest.staff-position = #-2
  R1
  \override MultiMeasureRest.staff-position = #0
  R1
  \override MultiMeasureRest.staff-position = #2
  R1
  \override MultiMeasureRest.staff-position = #3
  R1
  \override MultiMeasureRest.staff-position = #6
  R1
  \revert MultiMeasureRest.staff-position
  \break

  % In two Voices, odd-numbered voices are under the top line
  << { R1 } \\ { a1 } >>
  % Even-numbered voices are under the bottom line
  << { a1 } \\ { R1 } >>
  % Multi-measure rests in both voices remain separate
  << { R1 } \\ { R1 } >>

  % Separating multi-measure rests in more than two voices
  % requires an override
  << { R1 } \\ { R1 } \\
     \once \override MultiMeasureRest.staff-position = #0
     { R1 }
  >>

  % Using compressed bars in multiple voices requires another override
  % in all voices to avoid multiple instances being printed
  \compressFullBarRests
  <<
   \revert MultiMeasureRest.direction
    { R1*3 }
    \\
   \revert MultiMeasureRest.direction
    { R1*3 }
  >>
}
```



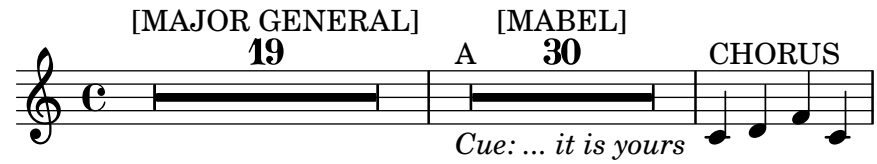


Testo a margine delle pause multiple

Il testo a margine di una pausa multipla viene centrato sopra o sotto di essa. Se il testo è lungo, la misura non si espanderà. Per espandere la pausa multipla in modo che si allinei col testo, conviene usare un accordo vuoto con del testo attaccato prima della pausa multipla.

Il testo così attaccato a una nota spaziatrice viene allineato a sinistra della posizione in cui la nota sarebbe posta nella misura, ma se la lunghezza della misura è determinata dalla lunghezza del testo, il testo verrà centrato.

```
\relative c' {
  \compressFullBarRests
  \textLengthOn
  <>^\markup { [MAJOR GENERAL] }
  R1*19
  <>_\markup { \italic { Cue: ... it is yours } }
  <>^\markup { A }
  R1*30^\markup { [MABEL] }
  \textLengthOff
  c4^\markup { CHORUS } d f c
}
```



Vedi anche

Glossario musicale: Sezione “pausa multipla” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Durations], pagina ⟨undefined⟩, ⟨undefined⟩ [Text], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MultiMeasureRest” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestNumber” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestText” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Se una diteggiatura viene posta su una pausa multipla (ad esempio `R1*10-4`), il numero della diteggiatura può collidere col numero del contatore delle battute.

Non è possibile condensare automaticamente molteplici pause normali in in una singola pausa multipla.
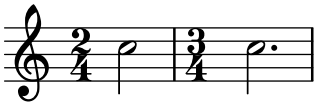
Le pause multiple non considerano le collisioni di pausa.

1.2.3 Aspetto dei ritmi

Indicazione di tempo

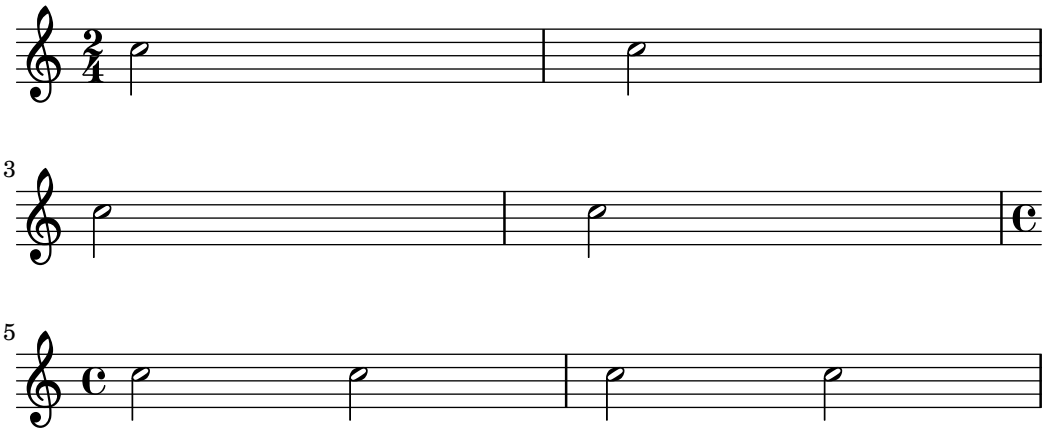
L’indicazione di tempo si imposta così:

```
\time 2/4 c2
\time 3/4 c2.
```
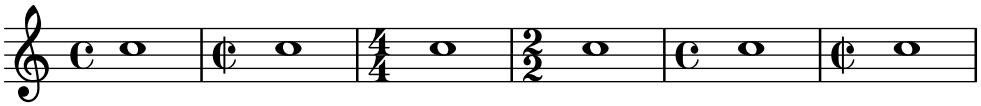


Le indicazioni di tempo appaiono all'inizio di un brano e ogni volta che l'indicazione cambia. Se il cambio ha luogo alla fine di un rigo, appare un'indicazione di tempo di precauzione. Si può modificare questo comportamento predefinito, come è spiegato in Sezione 5.4.6 [Visibility of objects], pagina 592.

```
\time 2/4
c2 c
\break
c c
\break
\time 4/4
c c c c
```



Il simbolo di indicazione di tempo usato nei tempi 2/2 e 4/4 può essere sostituito da un numero:

```
% Stile predefinito
\time 4/4 c1
\time 2/2 c1
% Passaggio allo stile numerico
\numericTimeSignature
\time 4/4 c1
\time 2/2 c1
% Ritorno allo stile predefinito
\defaultTimeSignature
\time 4/4 c1
\time 2/2 c1
```
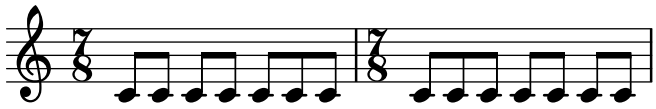


Le indicazioni di tempo mensurali sono trattate in [Mensural time signatures], pagina 419.

Oltre a impostare l'indicazione di tempo che appare nel pentagramma, il comando `\time` imposta anche i valori delle proprietà basate sull'indicazione di tempo, ovvero `baseMoment`, `beatStructure` e `beamExceptions`. I valori predefiniti di queste proprietà si trovano in '`scm/time-signature-settings.scm`'.
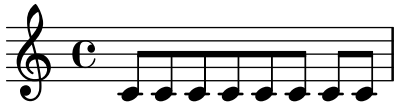
Si può sovrascrivere il valore predefinito di `beatStructure` nel comando `\time` stesso specificandolo come primo argomento opzionale:

```
\score {
  \new Staff {
    \relative c' {
      \time #'(2 2 3) 7/8
      \repeat unfold 7 { c8 } |
      \time #'(3 2 2) 7/8
      \repeat unfold 7 { c8 } |
    }
  }
}
```



Oppure si possono impostare tutti i valori predefiniti di queste variabili relative all'indicazione di tempo, incluse `baseMoment` e `beamExceptions`. I valori possono essere impostati in modo indipendente per diverse indicazioni di tempo. I nuovi valori hanno effetto appena viene eseguito un nuovo comando `\time` che abbia lo stesso valore dell'indicazione di tempo specificata nelle nuove impostazioni:

```
\score {
  \new Staff {
    \relative c' {
      \overrideTimeSignatureSettings
        4/4        % timeSignatureFraction
        1/4        % baseMomentFraction
        #'(3 1)    % beatStructure
        #'()       % beamExceptions
      \time 4/4
      \repeat unfold 8 { c8 } |
    }
  }
}
```
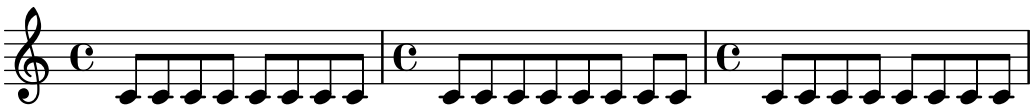


`\overrideTimeSignatureSettings` prende quattro argomenti:

1. *`timeSignatureFraction`*, una frazione che indica l'indicazione di tempo a cui questi valori si riferiscono.
2. *`baseMomentFraction`*, una frazione che contiene il numeratore e il denominatore dell'unità di tempo.
3. *`beatStructure`*, una lista Scheme che indica la struttura dei battiti nella misura, nell'unità di *`baseMomentFraction`*.

4. *beamExceptions*, una lista di associazione (*alist*) che contiene regole di disposizione delle travature che vanno oltre la fine ad ogni battito, come descritto in ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩.

I valori modificati delle proprietà predefinite dell'indicazione di tempo possono essere ripristinati ai valori originali:

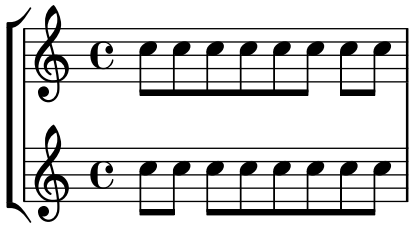
```
\score{
  \relative c' {
    \repeat unfold 8 { c8 } |
    \overrideTimeSignatureSettings
      4/4        % timeSignatureFraction
      1/4        % baseMomentFraction
      #'(3 1)    % beatStructure
      #'()       % beamExceptions
    \time 4/4
    \repeat unfold 8 { c8 } |
    \revertTimeSignatureSettings 4/4
    \time 4/4
    \repeat unfold 8 { c8 } |
  }
}
```



Si possono stabilire valori diversi delle proprietà predefinite dell'indicazione di tempo per righi diversi spostando `Timing_translator` e `Default_bar_line_engraver` dal contesto `Score` al contesto `Staff`.

```
\score {
  \new StaffGroup <<
     \new Staff {
        \overrideTimeSignatureSettings
          4/4        % timeSignatureFraction
          1/4        % baseMomentFraction
          #'(3 1)    % beatStructure
          #'()       % beamExceptions
        \time 4/4
        \repeat unfold 8 {c''8}
     }
     \new Staff {
        \overrideTimeSignatureSettings
          4/4        % timeSignatureFraction
          1/4        % baseMomentFraction
          #'(1 3)    % beatStructure
          #'()       % beamExceptions
        \time 4/4
        \repeat unfold 8 {c''8}
     }
  >>
  \layout {
    \context {
```

```
      \Score
      \remove "Timing_translator"
      \remove "Default_bar_line_engraver"
    }
    \context {
      \Staff
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    }
  }
}
```



Un ulteriore metodo per modificare queste variabili relative all'indicazione di tempo, che evita di mostrare di nuovo l'indicazione di tempo al momento del cambio, è descritto in ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩.

Comandi predefiniti

`\numericTimeSignature`, `\defaultTimeSignature`.
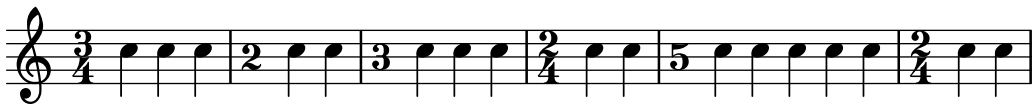
Frammenti di codice selezionati

Indicazione di tempo che mostra solo il numeratore (invece della frazione)

Talvolta un'indicazione di tempo non deve mostrare la frazione intera (ad esempio 7/4), ma solo il numeratore (7 in questo caso). Si può ottenere facilmente con `\override Staff.TimeSignature.style = #'single-digit`, che cambia lo stile in modo permanente. Con `\revert Staff.TimeSignature.style`, questa impostazione può essere annullata. Per applicare lo stile a cifra singola (`single-digit`) a una sola indicazione di tempo, si usa il comando `\override` preceduto da `\once`.

```
\relative c'' {
  \time 3/4
  c4 c c
  % Change the style permanently
  \override Staff.TimeSignature.style = #'single-digit
  \time 2/4
  c4 c
  \time 3/4
  c4 c c
  % Revert to default style:
  \revert Staff.TimeSignature.style
  \time 2/4
  c4 c
  % single-digit style only for the next time signature
  \once \override Staff.TimeSignature.style = #'single-digit
  \time 5/4
  c4 c c c c
```

```
  \time 2/4
  c4 c
}
```



Vedi anche

Glossario musicale: Sezione “indicazione di tempo” in *Glossario Musicale*

Guida alla notazione: [Mensural time signatures], pagina 419, ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩, ⟨undefined⟩ [Time administration], pagina ⟨undefined⟩.

File installati: ‘`scm/time-signature-settings.scm`’.
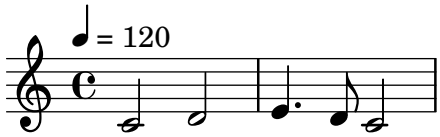
Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TimeSignature” in *Guida al Funzionamento Interno*, Sezione “Timing_translator” in *Guida al Funzionamento Interno*.

Indicazioni metronomiche
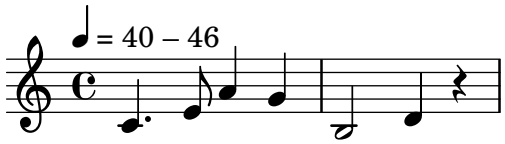
Un’indicazione metronomica è semplice da scrivere:

```
\tempo 4 = 120
c2 d
e4. d8 c2
```
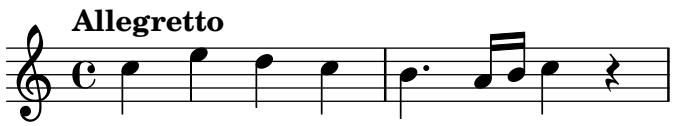


Le indicazioni metronomiche si possono rappresentare anche come una gamma di due numeri:

```
\tempo 4 = 40 - 46
c4. e8 a4 g
b,2 d4 r
```
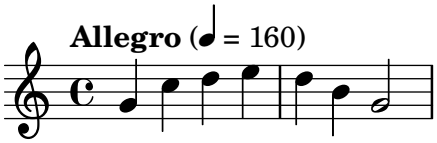


Al loro posto si possono usare delle indicazioni di tempo testuali:

```
\tempo "Allegretto"
c4 e d c
b4. a16 b c4 r4
```
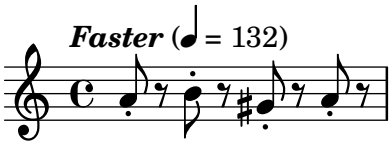


Un’indicazione metronomica, se combinata con del testo, viene posta automaticamente tra parentesi:

```
\tempo "Allegro" 4 = 160
g4 c d e
d4 b g2
```
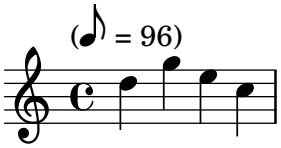


In generale, il testo può essere qualsiasi oggetto di tipo testuale:

```
\tempo \markup { \italic Faster } 4 = 132
a8-. r8 b-. r gis-. r a-. r
```
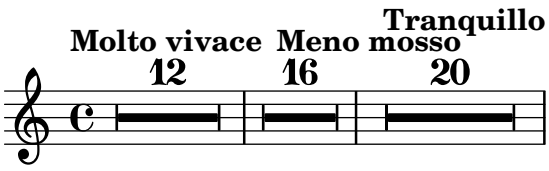


È possibile scrivere un'indicazione metronomica tra parentesi e senza testo includendo una stringa vuota nell'input:

```
\tempo "" 8 = 96
d4 g e c
```



In una parte per uno strumento che ha lunghi periodi pieni di pause, le indicazioni di tempo sono talvolta molto ravvicinate. Il comando `\markLengthOn` aggiunge dello spazio orizzontale per impedire che le indicazioni di tempo si sovrappongano; `\markLengthOff` ripristina il comportamento predefinito, per cui le indicazioni di tempo non sono tenute in considerazione ai fini della spaziatura orizzontale.

```
\compressFullBarRests
\markLengthOn
\tempo "Molto vivace"
R1*12
\tempo "Meno mosso"
R1*16
\markLengthOff
\tempo "Tranquillo"
R1*20
```



Frammenti di codice selezionati

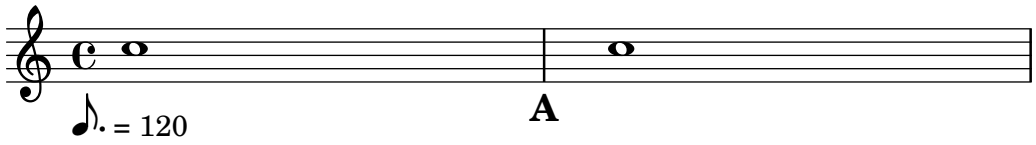
Posizionare il metronomo e i numeri di chiamata sotto il rigo

Di norma, il metronomo e i numeri di chiamata vengono posizionati sopra il rigo. Per metterli sotto il rigo basta impostare correttamente la proprietà `direction` di `MetronomeMark` o `RehearsalMark`.

```
\layout { ragged-right = ##f }

{
  % Metronome marks below the staff
  \override Score.MetronomeMark.direction = #DOWN
  \tempo 8. = 120
  c''1
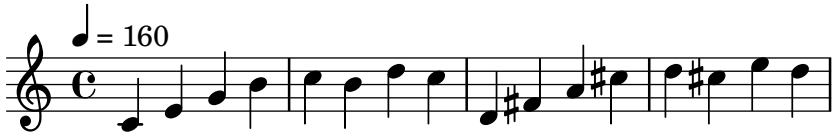
  % Rehearsal marks below the staff
  \override Score.RehearsalMark.direction = #DOWN
  \mark \default
  c''1
}
```



Modificare il tempo senza mostrare l'indicazione metronomica

Per cambiare il tempo del file MIDI senza che appaia l'indicazione metronomica, basta renderla invisibile.

```
\score {
  \new Staff \relative c' {
    \tempo 4 = 160
    c4 e g b
    c4 b d c
    \set Score.tempoHideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```
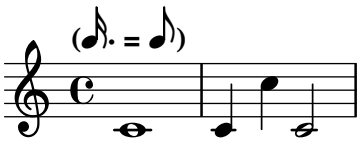


Creare indicazioni metronomiche in modalità testuale

Si possono creare nuove indicazioni metronomiche in modalità testuale, ma non modificheranno il tempo del file MIDI.

```
\relative c' {
  \tempo \markup {
```

```
      \concat {
        (
        \smaller \general-align #Y #DOWN \note #"16." #1
        " = "
        \smaller \general-align #Y #DOWN \note #"8" #1
        )
      }
    }
    c1
    c4 c' c,2
}
```



I dettagli si trovano in ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Vedi anche

Glossario musicale: Sezione “metronomo” in *Glossario Musicale*, Sezione “indicazione di tempo” in *Glossario Musicale*, Sezione “indicazione metronomica” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, Sezione 3.5 [MIDI output], pagina 491.

Frammenti di codice: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MetronomeMark” in *Guida al Funzionamento Interno*.
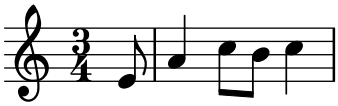
Anacrusi

Le misure parziali, come un *anacrusi* o una battuta in levare, si inseriscono col comando `\partial`,

```
\partial durata
```
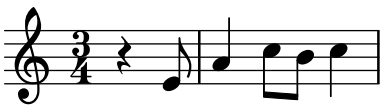
dove `durata` è la lunghezza *rimanente* della misura parziale *prima* dell’inizio della nuova misura completa.

```
\time 3/4
\partial 8
e8 | a4 c8 b c4 |
```



La *durata* può avere qualsiasi valore inferiore a quello di una misura intera:

```
\time 3/4
\partial 4.
r4 e8 | a4 c8 b c4 |
```
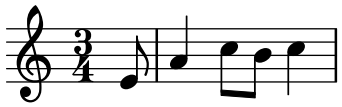


L’espressione `\partial durata` si può scrivere anche così:

```
\set Timing.measurePosition -durata
```

Quindi l'esempio precedente può essere scritto così:

```
\time 3/4
\set Timing.measurePosition = #(ly:make-moment -1/8)
e8 | a4 c8 b c4 |
```



La proprietà `measurePosition` contiene un numero razionale, solitamente positivo, che indica la durata corrispondente alla parte di misura trascorsa. Il comando `\partial durata` lo imposta su un numero negativo quando ha un senso diverso: in quel caso significa che la battuta corrente (la prima) sarà *preceduta* da una battuta 0 (la battuta parziale) della durata indicata da *durata*.

Vedi anche

Glossario musicale: Sezione "anacrusi" in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Grace notes], pagina ⟨undefined⟩.
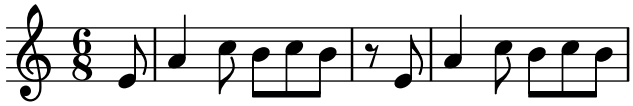
Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Timing_translator" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Il comando `\partial` deve essere collocato solo all'inizio di un brano. Se è posto dopo l'inizio, il programma potrebbe produrre degli avvisi o si potrebbero verificare problemi, dunque si consiglia di usare `\set Timing.measurePosition` al suo posto.
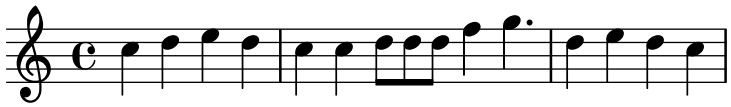
```
\time 6/8
\partial 8
e8 | a4 c8 b[ c b] |
\set Timing.measurePosition = #(ly:make-moment -1/4)
r8 e,8 | a4 c8 b[ c b] |
```



Musica in tempo libero
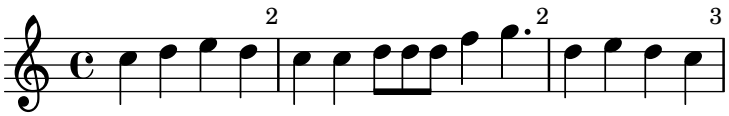
Nella musica in un tempo determinato l'inserimento delle stanghette e dei numeri di battuta è calcolato automaticamente. Nella musica in tempo libero (per esempio, la cadenza), un simile comportamento non è desiderabile, e può essere 'disabilitato' col comando `\cadenzaOn` e poi 'riabilitato' quando necessario con `\cadenzaOff`.

```
c4 d e d
\cadenzaOn
c4 c d8[ d d] f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```
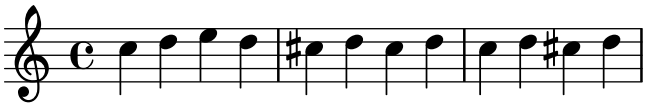


La numerazione delle battute riprende dopo la cadenza.

```
% Mostra tutti i numeri di battuta
\override Score.BarNumber.break-visibility = #all-visible
c4 d e d
\cadenzaOn
c4 c d8[ d d] f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```
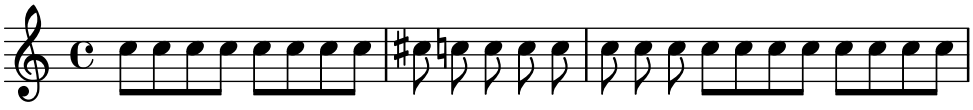


Se si inserisce un comando `\bar` dentro una cadenza non viene iniziata una nuova misura, anche se appare una stanghetta nell'output. Quindi qualsiasi alterazione, che di solito si considera sempre attiva fino alla fine della misura, sarà ancora valida dopo la stanghetta stampata da `\bar`. Se si desidera che le alterazioni successive appaiano, si dovranno inserire manualmente delle alterazioni forzate o di precauzione, come è spiegato in ⟨undefined⟩ [Accidentals], pagina ⟨undefined⟩.

```
c4 d e d
\cadenzaOn
cis4 d cis d
\bar "|"
% Il primo cis viene stampato senza alterazione anche se si trova dopo \bar
cis4 d cis! d
\cadenzaOff
\bar "|"
```



La disposizione automatica delle travature viene disabilitata da `\cadenzaOn`. Quindi tutte le travature nelle cadenze devono essere inserite manualmente. Si veda ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩.

```
\repeat unfold 8 { c8 }
\cadenzaOn
cis8 c c c c
\bar"|"
c8 c c
\cadenzaOff
\repeat unfold 8 { c8 }
```



Questi comandi predefiniti hanno effetto su tutti i righi di una partitura, anche quando inseriti in un solo contesto `Voice`. Per modificare questo comportamento, si sposta `Timing_translator` dal contesto `Score` al contesto `Staff`. Si veda ⟨undefined⟩ [Polymetric notation], pagina ⟨undefined⟩.

Comandi predefiniti

`\cadenzaOn`, `\cadenzaOff`.

Vedi anche

Glossario musicale: Sezione “cadenza” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.4.6 [Visibility of objects], pagina 592, ⟨undefined⟩ [Polymetric notation], pagina ⟨undefined⟩, ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Accidentals], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Problemi noti e avvertimenti

Le interruzioni automatiche di linea e di pagina possono aver luogo solo dopo una stanghetta di battuta; quindi, per consentire delle interruzioni nei lunghi passaggi di musica in tempo libero è necessario inserire manualmente delle stanghette ‘invisibili’:

```
\bar ""
```

Notazione polimetrica

La notazione polimetrica è supportata esplicitamente o tramite la modifica manuale del simbolo d’indicazione di tempo (e la trasformazione della durata delle note).

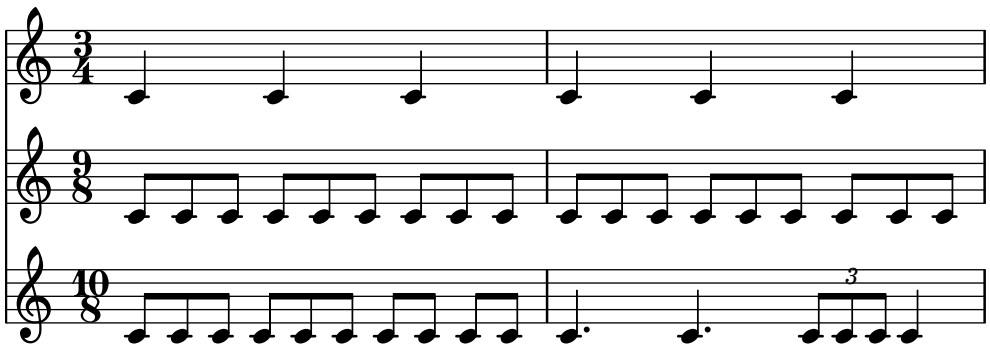
Diverse indicazioni di tempo con misure di uguale lunghezza

Si sceglie una normale indicazione di tempo per ogni rigo e si imposta `timeSignatureFraction` sulla frazione desiderata. Quindi si usa la funzione `\scaleDurations` per scalare la durata delle note di ogni rigo in modo che rientrino nella comune indicazione di tempo.

L’esempio seguente presenta simultaneamente musica con indicazioni di tempo di 3/4, 9/8 e 10/8. Nel secondo rigo le durate appaiono come moltiplicate per 2/3 (perché 2/3 * 9/8 = 3/4), mentre nel terzo rigo le durate appaiono come moltiplicate per 3/5 (perché 3/5 * 10/8 = 3/4). È possibile che si debbano inserire a mano le travature, perché la scalatura delle durate influenzerà le regole della disposizione automatica delle travature.

```
\relative c' <<
  \new Staff {
    \time 3/4
    c4 c c |
    c4 c c |
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = 9/8
    \scaleDurations 2/3
      \repeat unfold 6 { c8[ c c] }
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = 10/8
    \scaleDurations 3/5 {
      \repeat unfold 2 { c8[ c c] }
      \repeat unfold 2 { c8[ c] } |
      c4. c \tuplet 3/2 { c8[ c c] } c4
    }
  }
```

```
>>
```



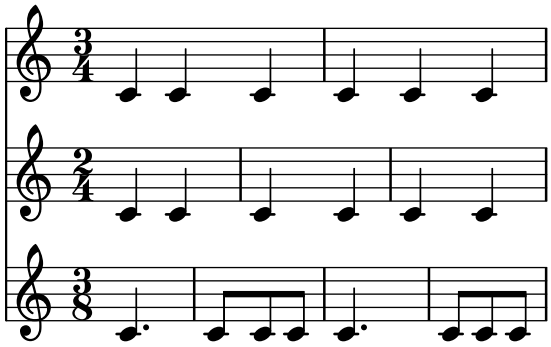
Diverse indicazioni di tempo con misure di lunghezza differenti

Si può dare a ogni rigo la sua indicazione di tempo indipendente spostando `Timing_translator` e `Default_bar_line_engraver` nel contesto `Staff`.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
}

% Ora ogni rigo ha la sua indicazione di tempo.

\relative c' <<
  \new Staff {
    \time 3/4
    c4 c c |
    c4 c c |
  }
  \new Staff {
    \time 2/4
    c4 c |
    c4 c |
    c4 c |
  }
  \new Staff {
    \time 3/8
    c4. |
    c8 c c |
    c4. |
    c8 c c |
  }
>>
```
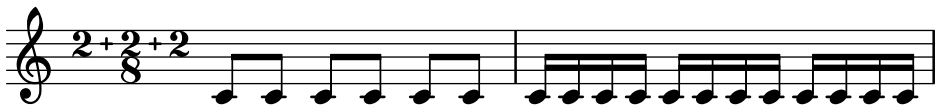


Indicazioni di tempo composto

Si creano con la funzione `\compoundMeter`. La sintassi è:

```
\compoundMeter #'(lista di liste)
```

La struttura più semplice è una singola lista, dove l'*ultimo* numero indica il numero inferiore dell'indicazione di tempo e i numeri precedenti indicano i numeri superiori del segno di tempo.

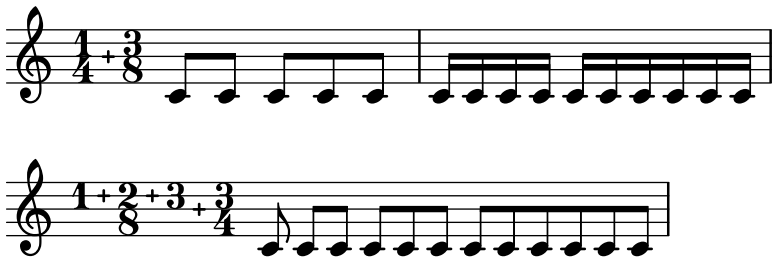
```
\relative c' {
  \compoundMeter #'((2 2 2 8))
  \repeat unfold 6 c8 \repeat unfold 12 c16
}
```



Si possono costruire tempi più complessi tramite ulteriori liste. Le modalità di disposizione automatica delle travature varieranno a seconda di questi valori.

```
\relative c' {
  \compoundMeter #'((1 4) (3 8))
  \repeat unfold 5 c8 \repeat unfold 10 c16
}

\relative c' {
  \compoundMeter #'((1 2 3 8) (3 4))
  \repeat unfold 12 c8
}
```



Vedi anche

Glossario musicale: Sezione “polimetrico” in *Glossario Musicale*, Sezione “indicazione di tempo polimetrico” in *Glossario Musicale*, Sezione “tempo” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Automatic beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩, ⟨undefined⟩ [Scaling durations], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TimeSignature” in *Guida al Funzionamento Interno*, Sezione “Timing_translator” in *Guida al Funzionamento Interno*, Sezione “Default_bar_line_engraver” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti
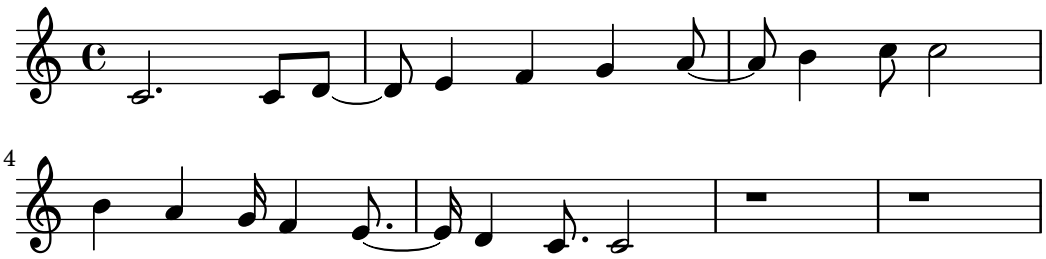
Quando si usano simultaneamente indicazioni di tempo diverse, le note che procedono parallelamente saranno poste nella stessa posizione sull’asse orizzontale. Tuttavia le stanghette dei vari righi faranno sì che la spaziatura delle note sia meno regolare in ciascun rigo di quanto accadrebbe normalmente senza le diverse indicazioni di tempo.

Divisione automatica delle note

Le note le cui durate eccedono il valore della battuta possono essere convertite automaticamente in note con legature di valore a cavallo delle stanghette sostituendo l’incisore `Note_heads_engraver` con `Completion_heads_engraver`. Analogalmente, le pause le cui durate eccedono il valore della battuta possono essere divise automaticamente sostituendo `Rest_engraver` con `Completion_rest_engraver`. Nell’esempio seguente, le note e le pause che eccedono la durata di battuta vengono divise e le note sono anche connesse con legature di valore a cavallo della stanghetta.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
  \remove "Rest_engraver"
  \consists "Completion_rest_engraver"
}

{ c2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2 }
```



Questi incisori dividono tutte le note e le pause in corrispondenza della stanghetta e inseriscono le legature di valore. Uno dei suoi usi possibili è la verifica di partiture complesse: se le misure non sono riempite interamente, le legature di valore mostrano esattamente di quanto è ecceduta ogni misura.

Vedi anche

Glossario musicale: Sezione “legatura di valore” in *Glossario Musicale*

Manuale di apprendimento: Sezione “Engravers explained” in *Manuale di Apprendimento*, Sezione “Adding and removing engravers” in *Manuale di Apprendimento*.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Completion_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Rest_engraver” in *Guida al Funzionamento Interno*, Sezione “Completion_rest_engraver” in *Guida al Funzionamento Interno*, Sezione “Forbid_line_break_engraver” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

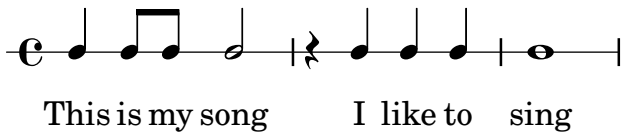
Non tutte le durate (specialmente quelle che contengono gruppi irregolari) possono essere rappresentate esattamente con normali note e punti, ma l'incisore `Completion_heads_engraver` non inserirà gruppi irregolari.

L'incisore `Completion_heads_engraver` ha effetto solo sulle note, non divide le pause.

Mostrare i ritmi della melodia

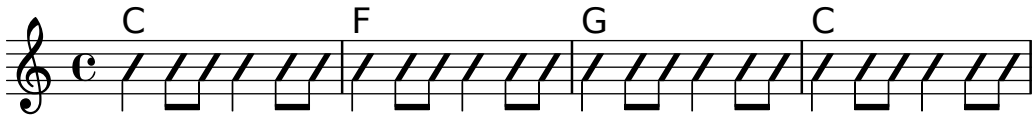
È possibile mostrare soltanto il ritmo di una melodia usando il rigo ritmico. Tutte le altezze delle note su tale rigo sono appiattite e il rigo stesso ha una sola linea

```
<<
  \new RhythmicStaff {
    \new Voice = "myRhythm" {
      \time 4/4
      c4 e8 f g2
      r4 g g f
      g1
    }
  }
  \new Lyrics {
    \lyricsto "myRhythm" {
      This is my song
      I like to sing
    }
  }
>>
```



I diagrammi degli accordi per chitarra di solito mostrano i ritmi di accompagnamento. Si possono visualizzare usando l'incisore `Pitch_squash_engraver` e il comando `\improvisationOn`.

```
<<
  \new ChordNames {
    \chordmode {
      c1 f g c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } \relative c'' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c
  }
>>
```



Comandi predefiniti

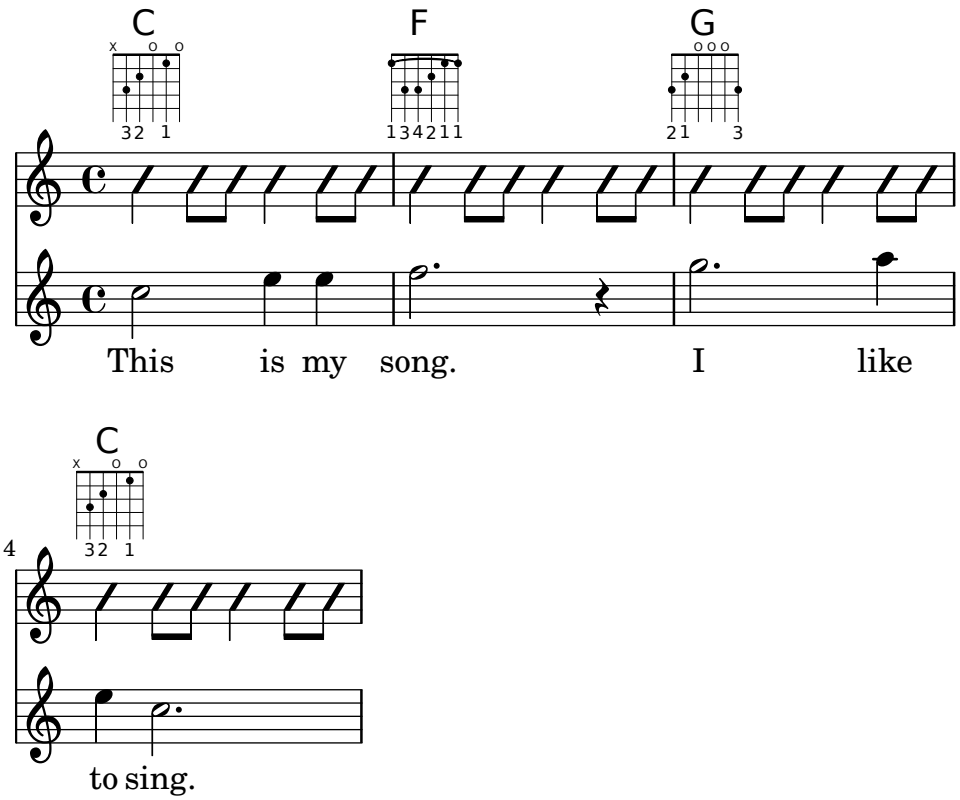
`\improvisationOn`, `\improvisationOff`.

Frammenti di codice selezionati

Ritmi di accompagnamento per chitarra

Per la musica per chitarra, è possibile mostrare i ritmi di accompagnamento, insieme alle note della melodia e ai nomi e ai diagrammi degli accordi.

```
\include "predefined-guitar-fretboards.ly"
<<
  \new ChordNames {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new FretBoards {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } {
    \relative c'' {
      \improvisationOn
      c4 c8 c c4 c8 c
      f4 f8 f f4 f8 f
      g4 g8 g g4 g8 g
      c4 c8 c c4 c8 c
    }
  }
  \new Voice = "melody" {
    \relative c'' {
      c2 e4 e4
      f2. r4
      g2. a4
      e4 c2.
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      This is my song.
      I like to sing.
    }
  }
>>
```



Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

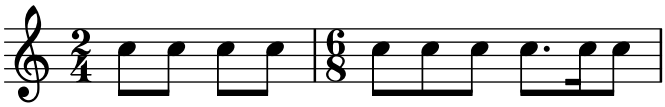
Guida al funzionamento interno: Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*, Sezione “Pitch_squash_engraver” in *Guida al Funzionamento Interno*.

1.2.4 Travature

Travature automatiche

Le travature sono inserite automaticamente:
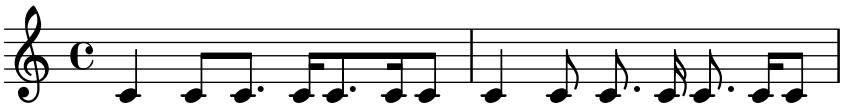
```
\time 2/4 c8 c c c
\time 6/8 c8 c c c8. c16 c8
```



Se queste impostazioni automatiche non sono soddisfacenti, si può definire esplicitamente la disposizione delle travature, come è spiegato in ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩. Le travature *devono* essere inserite manualmente se devono estendersi oltre le pause.

La disposizione automatica delle travature, se non necessaria, può essere disabilitata con `\autoBeamOff` e riabilitata con `\autoBeamOn`:

```
c4 c8 c8. c16 c8. c16 c8
\autoBeamOff
c4 c8 c8. c16 c8.
\autoBeamOn
c16 c8
```



Nota: Se si usano le travature per indicare i melismi nelle parti vocali, occorre disabilitare la disposizione automatica delle travature con `\\autoBeamOff` e le travature devono essere indicate manualmente. L'uso di `\partcombine` insieme a `\autoBeamOff` può produrre risultati imprevisti. Si vedano i frammenti di codice per avere maggiori informazioni.

Si possono creare dei modelli di disposizione delle travature diversi da quelli automatici predefiniti, come è spiegato in ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩.

Comandi predefiniti
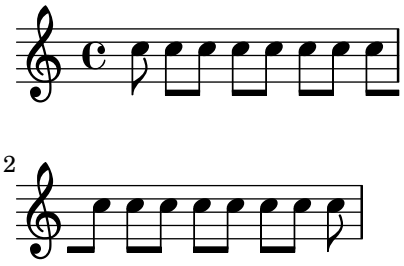
`\autoBeamOff`, `\autoBeamOn`.

Frammenti di codice selezionati

Travature che attraversano le interruzioni di linea

Le interruzioni di linea sono di norma proibite quando le travature attraversano la stanghetta di una battuta. Si può cambiare questo comportamento nel modo seguente:
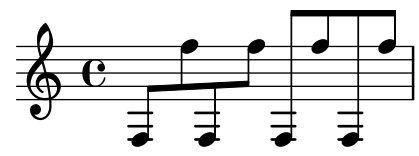
```
\relative c'' {
  \override Beam.breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
  c8] c[ c] c[ c] c[ c] c
}
```



Modificare la distanza delle travature angolari

Le travature angolari vengono inserite automaticamente quando viene rilevata un'ampia distanza tra le teste di nota. Questo comportamento può essere regolato attraverso la proprietà `auto-knee-gap`. Viene disegnata una travatura angolare se la distanza è più grande del valore di `auto-knee-gap` più la larghezza della travatura (che dipende dalla durata delle note e dall'inclinazione della travatura). Il valore predefinito di `auto-knee-gap` è 5.5 spazi rigo.

```
{
  f8 f''8 f8 f''8
  \override Beam.auto-knee-gap = #6
  f8 f''8 f8 f''8
}
```



Partcombine e autoBeamOff

La funzione `\autoBeamOff`, se usata insieme a `\partcombine`, può essere difficile da comprendere.

È preferibile usare invece

`\set Staff.autoBeaming = ##f`
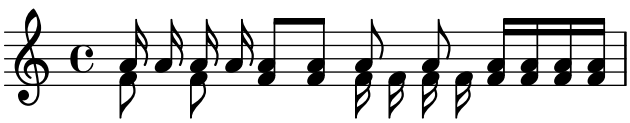
per assicurarsi che la disposizione delle travature sia disabilitata per tutto il rigo.

`\partcombine` funziona con 3 voci – gambo in su singolo, gambo in giù singolo, gambo in su unito.

L'uso di `\autoBeamOff` all'interno del primo argomento di partcombine ha effetto sulla voce che è attiva al momento in cui la funzione viene elaborata, ovvero sul gambo in su singolo o sul gambo in giù unito. L'uso di `\autoBeamOff` nel secondo argomento avrà effetto sulla voce che ha il gambo in giù singolo.

Per poter usare `\autoBeamOff` per impedire tutte le disposizioni automatiche delle travature, se usato con `\partcombine`, è necessario richiamare tre volte la funzione `\autoBeamOff`.

```
{
  %\set Staff.autoBeaming = ##f % turns off all autobeaming
  \partcombine
  {
    \autoBeamOff % applies to split up stems
    \repeat unfold 4 a'16
    %\autoBeamOff % applies to combined up stems
    \repeat unfold 4 a'8
    \repeat unfold 4 a'16
  }
  {
    \autoBeamOff % applies to down stems
    \repeat unfold 4 f'8
    \repeat unfold 8 f'16 |
  }
}
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩.

File installati: '`scm/auto-beam.scm`'.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Auto_beam_engraver" in *Guida al Funzionamento Interno*, Sezione "Beam_engraver" in *Guida al Funzionamento Interno*, Sezione "Beam" in *Guida al Funzionamento Interno*, Sezione "BeamEvent" in *Guida al Funzionamento Interno*, Sezione "BeamForbidEvent" in *Guida al Funzionamento Interno*, Sezione "beam-interface" in *Guida al Funzionamento Interno*, Sezione "unbreakable-spanner-interface" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Le proprietà di una travatura sono determinate all'*inizio* della sua costruzione e qualsiasi ulteriore modifica alle sue proprietà che venga fatta prima che la travatura sia stata completata non avrà effetto finché non inizia la *successiva*, nuova travatura.

Impostare il comportamento delle travature automatiche

Quando la disposizione automatica delle travature è abilitata, la disposizione delle travature è determinata da tre proprietà di contesto: `baseMoment`, `beatStructure` e `beamExceptions`. I valori predefiniti di queste variabili possono essere sovrascritti, come vedremo tra breve, oppure si possono anche cambiare i valori predefiniti stessi, come è spiegato in ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.

Se è definita una regola `beamExceptions` per l'indicazione di tempo corrente, tale regola soltanto determina la disposizione delle travature; i valori di `baseMoment` e `beatStructure` vengono ignorati. Se non è definita alcuna regola `beamExceptions` per l'indicazione di tempo corrente, la disposizione delle travature è determinata dai valori di `baseMoment` e `beatStructure`.
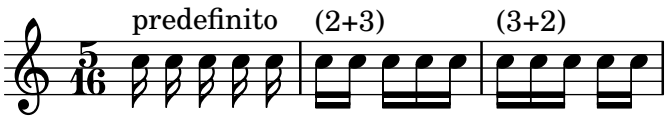
Disposizione delle travature basata su `baseMoment` *e* `beatStructure`

Dato che le indicazioni di tempo più comuni hanno delle regole `beamExceptions` già definite, occorre disabilitarle se la disposizione automatica deve basarsi su `baseMoment` e `beatStructure`. Le regole `beamExceptions` si disabilitano con questo comando

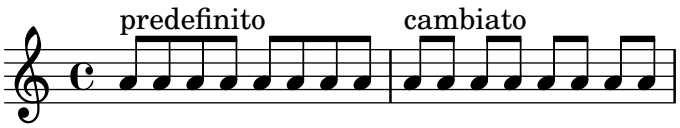
```
\set Timing.beamExceptions = #'()
```

Quando `beamExceptions` è impostato su `#'()`, o per impostazione esplicita o perché non sono state definite internamente le `beamExceptions` per l'indicazione di tempo corrente, le estremità delle travature si trovano sulle suddivisioni come specificato dalle proprietà di contesto `baseMoment` e `beatStructure`. `beatStructure` è una lista *scheme* che definisce la lunghezza di ogni suddivisione in rapporto alla misura in unità di `baseMoment`. Per impostazione predefinita, `baseMoment` è uno fratto il denominatore dell'indicazione di tempo e ogni unità di `baseMoment` corrisponde a una singola suddivisione.

```
\time 5/16
c16^"predefinito" c c c c |
% È improbabile che per un tempo di 5/16 sia stata definita beamExceptions,
% ma disabilitiamola lo stesso per sicurezza
\set Timing.beamExceptions = #'()
\set Timing.beatStructure = #'(2 3)
c16^"(2+3)" c c c c |
\set Timing.beatStructure = #'(3 2)
c16^"(3+2)" c c c c |
```



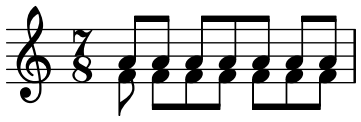

```
\time 4/4
a8^"predefinito" a a a a a a a
% Disabilita beamExceptions perché è senz'altro definita
% per il tempo 4/4
\set Timing.beamExceptions = #'()
\set Timing.baseMoment = #(ly:make-moment 1/4)
\set Timing.beatStructure = #'(1 1 1 1)
a8^"cambiato" a a a a a a a
```



Le modifiche alle impostazioni delle travature possono essere limitate a contesti specifici. Se non si specifica alcuna impostazione in un contesto di livello più basso, verrà applicata l'impostazione del contesto che lo contiene.

```
\new Staff {
  \time 7/8
  % Nessun bisogno di disabilitare beamExceptions perché non è definita per il tempo 7/
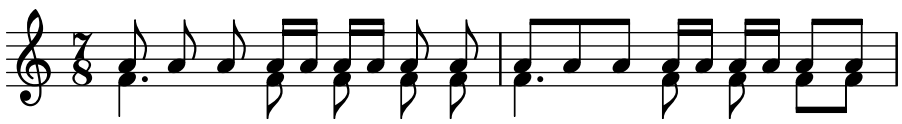
  \set Staff.beatStructure = #'(2 3 2)
  <<
    \new Voice = one {
      \relative c'' {
        a8 a a a a a a
      }
    }
    \new Voice = two {
      \relative c' {
        \voiceTwo
        \set Voice.beatStructure = #'(1 3 3)
        f8 f f f f f f
      }
    }
  >>
}
```



Quando si usano più voci, occorre specificare il contesto `Staff` se si vuole applicare la disposizione delle travature a tutte le voci del rigo:

```
\time 7/8
% ritmo 3-1-1-2
% Se non si specifica il contesto, la modifica viene applicata a Voice e quindi non fun
% Dato che le voci sono autogenerate, tutto il ritmo avrà come baseMoment (1 . 8)
\set beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \\ {f4. f8 f f f} >>
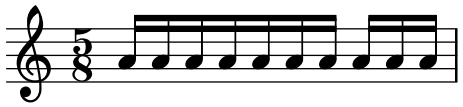
% Funziona correttamente se si specifica il contesto Staff
\set Staff.beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \\ {f4. f8 f f f} >>
```



Il valore di `baseMoment` può essere regolato in modo da cambiare il comportamento delle travature, se si vuole. In questo caso occorre cambiare anche il valore di `beatStructure` così che sia compatibile col nuovo valore di `baseMoment`.

```
\time 5/8
% Nessun bisogno di disabilitare beamExceptions perché non è definita per il tempo 5/8
\set Timing.baseMoment = #(ly:make-moment 1/16)
```

```
\set Timing.beatStructure = #'(7 3)
\repeat unfold 10 { a16 }
```
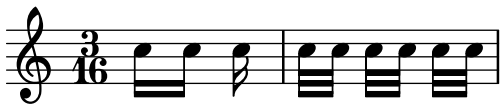


`baseMoment` è un *momento*, ovvero un'unità della durata musicale. Una quantità di tipo *moment* viene creata dalla funzione scheme `ly:make-moment`. Per maggiori informazioni su questa funzione, si veda ⟨undefined⟩ [Time administration], pagina ⟨undefined⟩.

Per impostazione predefinita, `baseMoment` ha un valore di uno fratto il denominatore dell'indicazione di tempo. Le eccezioni a questa regola si trovano in '`scm/time-signature-settings.scm`'.

Disposizione delle travature con `beamExceptions`

Le regole speciali di disposizione automatica delle travature (diverse da quelle che determinano la corrispondenza della travatura alla suddivisione) sono definite nella proprietà `beamExceptions`.

```
\time 3/16
\set Timing.beatStructure = #'(2 1)
\set Timing.beamExceptions =
  #'(                          ;inizio della lista di associazioni (alist)
     (end .                    ;estremità delle travature
      (                        ;inizio della lista che indica gli estremi
       ((1 . 32) . (2 2 2))    ;regola per le travature da 1/32 -- termina ognuna a 1/16
      )))                      %chiude tutti gli elementi
c16 c c |
\repeat unfold 6 { c32 } |
```



`beamExceptions` è una lista di associazioni (*alist*) che ha una chiave che indica il tipo di regola e un valore che esprime le regole di disposizione delle travature.

Al momento l'unico tipo di regola disponibile è `'end`, che specifica il termine della travatura.

Le regole di disposizione delle travature sono una lista di associazione scheme (o lista di coppie) che indica il tipo di travatura e la modalità di raggruppamento da applicare alle travature contenenti note dalla durata più breve del tipo di travatura a loro assegnato.

```
#'((travatura-1 . raggruppamento-1)
   (travatura-2 . raggruppamento-2)
   (travatura-3 . raggruppamento-3))
```

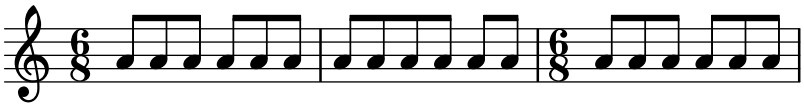
Il tipo di travatura è una coppia scheme che indica la durata della travatura, ad esempio `(1 . 16)`.

Il raggruppamento è una lista scheme che indica il raggruppamento da usare per la travatura; ha come unità la durata specificata nel tipo di travatura.

Nota: Il valore di `beamExceptions` deve essere una lista *completa* di eccezioni, ovvero bisogna includere tutte le eccezioni che si vogliono applicare. Non è possibile aggiungere, rimuovere o modificare soltanto una eccezione. Anche se questo può sembrare scomodo, significa anche che non c'è bisogno di conoscere le attuali impostazioni delle travature per poter specificare un nuovo modello di disposizione delle travature.

Quando cambia l'indicazione di tempo, vengono impostati i valori predefiniti di `Timing.baseMoment`, `Timing.beatStructure` e `Timing.beamExceptions`. L'impostazione dell'indicazione di tempo ripristina le impostazioni automatiche delle travature del contesto `Timing` ai valori predefiniti.
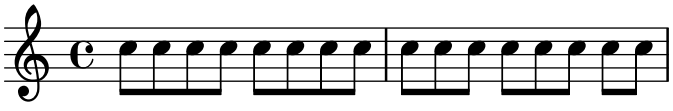
```
\time 6/8
\repeat unfold 6 { a8 }
% raggruppamento (4 + 2)
\set Timing.beatStructure = #'(4 2)
\repeat unfold 6 { a8 }
% ritorno al comportamento predefinito
\time 6/8
\repeat unfold 6 { a8 }
```



Le impostazioni predefinite della disposizione automatica delle travature per ogni tempo sono definite in '`scm/time-signature-settings.scm`'. La loro modifica è descritta in ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.

Molte impostazioni di travature automatiche per le indicazioni di tempo hanno un elemento `beamExceptions`. Ad esempio, il tempo 4/4 cerca di creare due travature nella misura se ci sono solo note di un ottavo. La regola `beamExceptions` può sovrascrivere l'impostazione di `beatStructure` se `beamExceptions` non viene annullato.

```
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = #'(3 3 2)
% Le travature non saranno raggruppate in (3 3 2) a causa di beamExceptions
\repeat unfold 8 {c8} |
% Il raggruppamento delle travature è (3 3 2) perché abbiamo tolto le impostazioni prec
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c8}
```
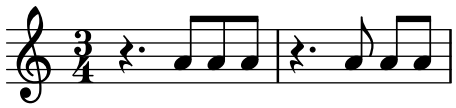


Analogalmente, le note di un ottavo in un tempo 3/4 sono raggruppate in un'unica travatura. Per raggrupparle secondo le suddivisioni, azzera `beamExceptions`.

```
\time 3/4
% il comportamento predefinito è un gruppo di (6) a causa di beamExceptions
\repeat unfold 6 {a8} |
% Le travature saranno raggruppate in (1 1 1) a causa dei valori predefiniti di baseMor
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a8}
```



Spesso, nelle partiture di età classica e romantica, le travature iniziano a metà della misura in un tempo 3/4; ma la pratica moderna preferisce evitare l'impressione ingannevole di un tempo 6/8 (vedi Gould, p. 153). Situazioni simili si incontrano anche per il tempo 3/8. Questo comportamento è controllato dalla proprietà di contesto `beamHalfMeasure`, che ha effetto soltanto sulle indicazioni di tempo che hanno 3 come numeratore:

```
\time 3/4
r4. a8 a a |
\set Timing.beamHalfMeasure = ##f
r4. a8 a a |
```



Come funziona la disposizione automatica delle travature

Quando la disposizione automatica delle travature è abilitata, la disposizione delle travature è determinata dalle proprietà di contesto `baseMoment`, `beatStructure` e `beamExceptions`.

Nel determinare l'aspetto delle travature vengono applicate le seguenti regole, in ordine di priorità:

- Se si specifica una travatura manuale con `[...]` imposta la travatura in quel modo, altrimenti
- se è definita una regola di fine della travatura in `beamExceptions` per il tipo di travatura in questione, la usa per determinare i punti corretti in cui le travature possono terminare, altrimenti
- se è definita una regola di fine della travatura in `beamExceptions` per un tipo di travatura più lunga, la usa per determinare i punti corretti in cui le travature possono terminare, altrimenti
- usa i valori di `baseMoment` e `beatStructure` per determinare l'estensione delle suddivisioni della misura e terminare le travature alla fine delle suddivisioni.

Nelle regole precedenti, il *tipo di travatura* è la durata della nota più corta nel gruppo della travatura.

Le regole predefinite per le travature si trovano in '`scm/time-signature-settings.scm`'.

Frammenti di codice selezionati
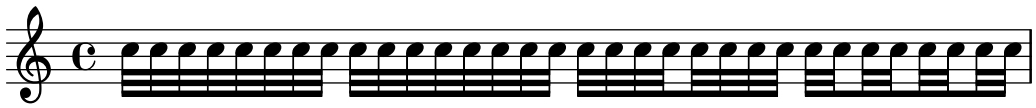
Suddividere le travature

Le travature di note consecutive di un sedicesimo (o più brevi) non vengono suddivise, ovvero i tre (o più) tratti della travatura si estendono, senza spezzarsi, sugli interi gruppi di note. Questo comportamento può essere modificato in modo da suddividere le travature in sottoraggruppamenti attraverso la proprietà `subdivideBeams`. Se impostata, le travature che comprendono più sottoraggruppamenti verranno suddivise a intervalli definiti dal valore attuale di `baseMoment`, riducendo le travature multiple a una sola travatura che collega i sottoraggruppamenti. Si noti che `baseMoment`, se non impostata esplicitamente, equivale a uno fratto il denominatore dell'attuale indicazione di tempo. Deve quindi essere impostata su una frazione che stabilisca la durata del sottogruppo di travature; lo si può fare usando la funzione `ly:make-moment`, come è mostrato in questo frammento di codice. Inoltre quando `baseMoment` cambia, anche `beatStructure` deve essere modificato per accordarsi con `baseMoment`:

```
\relative c'' {
  c32[ c c c c c c c]
  \set subdivideBeams = ##t
  c32[ c c c c c c c]

  % Set beam sub-group length to an eighth note
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = #'(2 2 2 2)
  c32[ c c c c c c c]
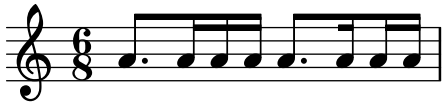
  % Set beam sub-group length to a sixteenth note
  \set baseMoment = #(ly:make-moment 1/16)
  \set beatStructure = #'(4 4 4 4)
  c32[ c c c c c c c]
}
```



Travatura che segue strettamente il battito

Si possono impostare i tratti di suddivisione della travatura in modo che siano rivolti verso la relativa pulsazione. La prima travatura fa sì che non spuntino i tratti di suddivisione (comportamento predefinito); la seconda travatura è orientata verso la pulsazione.

```
\relative c'' {
  \time 6/8
  a8. a16 a a
  \set strictBeatBeaming = ##t
  a8. a16 a a
}
```



Segni per la conduzione, segni di raggruppamento della misura

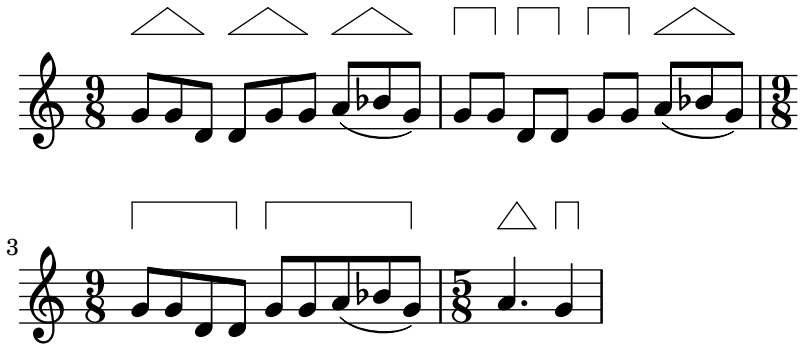
Il raggruppamento delle pulsazioni all'interno della misura è regolato dalla proprietà di contesto `beatStructure`. I valori di `beatStructure` per varie indicazioni di tempo vengono stabiliti in '`scm/time-signature-settings.scm`'. Questi valori possono essere impostati o modificati con `\set`. Altrimenti, si può usare `\time` per impostare sia l'indicazione di tempo che la struttura delle pulsazioni. Per farlo si specifica il raggruppamento interno delle pulsazioni in una misura in una lista di numeri (nella sintassi di Scheme) prima dell'indicazione di tempo.

`\time` agisce nel contesto `Timing`, dunque non reimposterà i i valori di `beatStructure` e `baseMoment` che sono impostati in altri contesti di più basso livello, come `Voice`.

Se si include l'incisore `Measure_grouping_engraver` in uno dei contesti che regolano l'aspetto, appariranno i segni di raggruppamento della misura. Tali segni facilitano la lettura di musica moderna ritmicamente complessa. Nell'esempio la misura di 9/8 è raggruppata in due diversi schemi usando due metodi differenti, mentre la misura di 5/8 è raggruppata in base alle impostazioni predefinite in '`scm/time-signature-settings.scm`':

```
\score {
  \new Voice \relative c'' {
```

```
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = #'(2 2 2 3)
    g8 g d d g g a( bes g) |
    \time #'(4 5) 9/8
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```
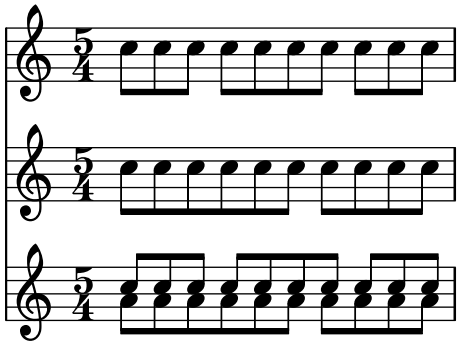


Estremità delle travature nel contesto Score

Le regole relative alle estremità delle travature definite nel contesto `Score` si applicano a tutti i righi, ma possono essere modificate anche ai livelli `Staff` e `Voice`:

```
\relative c'' {
  \time 5/4
  % Set default beaming for all staves
  \set Score.baseMoment = #(ly:make-moment 1/8)
  \set Score.beatStructure = #'(3 4 3)
  <<
    \new Staff {
      c8 c c c c c c c c c
    }
    \new Staff {
      % Modify beaming for just this staff
      \set Staff.beatStructure = #'(6 4)
      c8 c c c c c c c c c
    }
    \new Staff {
      % Inherit beaming from Score context
      <<
        {
          \voiceOne
          c8 c c c c c c c c c
        }
```

```
        % Modify beaming for this voice only
        \new Voice {
          \voiceTwo
          \set Voice.beatStructure = #'(6 4)
          a8 a a a a a a a a
        }
      >>
    }
  >>
}
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.

File installati: ‘`scm/time-signature-settings.scm`’.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Auto_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamForbidEvent” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*.

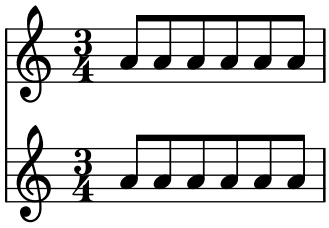
Problemi noti e avvertimenti

Se una partitura finisce prima del termine di una travatura automatica, cui mancano ancora delle note, quest’ultima travatura non apparirà. Lo stesso vale per le voci polifoniche, inserite con `<< ... \\ ... >>`. Una voce polifonica non apparirà se termina quando una travatura automatica è ancora in attesa di note. Per aggirare questi problemi occorre impostare manualmente l’ultima travatura della voce o della partitura.

`Timing` è un alias del contesto `Score`. Questo significa che la modifica della disposizione delle travature in un rigo avrà effetto anche sugli altri righi. Quindi un’impostazione di tempo in un rigo successivo reimposterà la disposizione personalizzata delle travature definita in un rigo precedente. Per evitare questo problema si può impostare l’indicazione di tempo su un solo rigo.

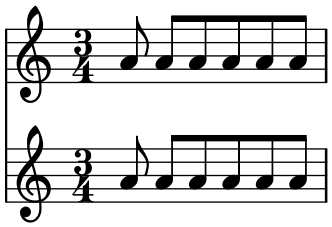
```
<<
  \new Staff {
    \time 3/4
    \set Timing.baseMoment = #(ly:make-moment 1/8)
    \set Timing.beatStructure = #'(1 5)
    \repeat unfold 6 { a8 }
  }
  \new Staff {
    \repeat unfold 6 { a8 }
  }
```

```
>>
```



Si possono cambiare anche le impostazioni predefinite delle travature, in modo che sia usata sempre la disposizione delle travature desiderata. Le modifiche nelle impostazioni della travatura automatica per le indicazioni di tempo sono descritte in ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.
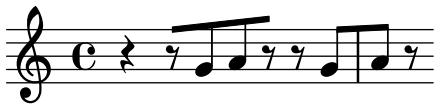
```
<<
  \new Staff {
    \overrideTimeSignatureSettings
      3/4       % timeSignatureFraction
      1/8       % baseMomentFraction
      #'(1 5)   % beatStructure
      #'()              % beamExceptions
    \time 3/4
    \repeat unfold 6 { a8 }
  }
  \new Staff {
    \time 3/4
    \repeat unfold 6 { a8 }
  }
>>
```



Travature manuali
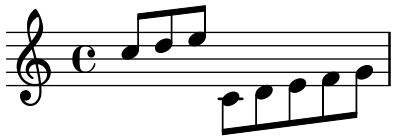
In alcuni casi potrebbe essere necessario scavalcare l'algoritmo di disposizione automatica delle travature. Ad esempio, questo algoritmo non inserirà delle travature tra le pause o tra le stanghette; e nelle partiture corali la disposizione delle travature è spesso determinato dall'articolazione del testo piuttosto che da quella musicale. Tali travature possono essere specificate manualmente indicandone l'inizio e la fine con `[` e `]`.

```
r4 r8[ g' a r] r g[ | a] r
```
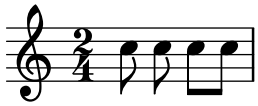


La direzione delle travature può essere impostata manualmente attraverso gli indicatori di direzione:

```
c8^[ d e] c,_[ d e f g]
```



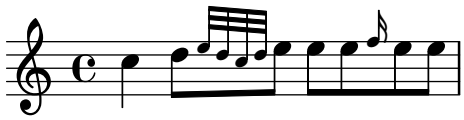
Le note individuali possono essere contrassegnate con `\noBeam` per impedire che vengano inserite in una travatura:

```
\time 2/4
c8 c\noBeam c c
```
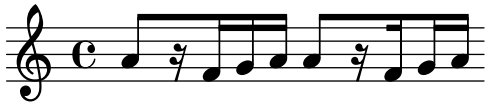


Le travature degli abbellimenti e quelle delle note normali possono coesistere simultaneamente. Gli abbellimenti privi di travatura non vengono inseriti nella travatura delle note normali.

```
c4 d8[
\grace { e32 d c d }
e8] e[ e
\grace { f16 }
e8 e]
```



Si può ottenere un controllo manuale delle travature ancora più preciso agendo sulle proprietà `stemLeftBeamCount` e `stemRightBeamCount`, che specificano il numero di travature da creare a sinistra e a destra della nota successiva. Se una di queste proprietà viene impostata, il suo valore verrà usato una volta sola, e la proprietà sarà poi cancellata. In questo esempio, l'ultima nota `f` ha una sola travatura a sinistra: la travatura corrispondente alla sottodivisione di un ottavo all'interno dell'intero raggruppamento.

```
a8[ r16 f g a]
a8[ r16
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #1
f16
\set stemLeftBeamCount = #1
g16 a]
```



Comandi predefiniti

`\noBeam`.

Frammenti di codice selezionati

Code e punte delle travature

È possibile ottenere delle codette su note isolate e dei tratti di suddivisione all'estremità della travatura con una combinazione di `stemLeftBeamCount`, `stemRightBeamCount` e una coppia di indicatori della travatura `[]`.

Per ottenere delle codette rivolte a destra, si usa la coppia di indicatori `[]` e si imposta `stemLeftBeamCount` a zero (vedi Example 1).

Per ottenere delle codette rivolte a sinistra, si imposta invece `stemRightBeamCount` (Example 2).

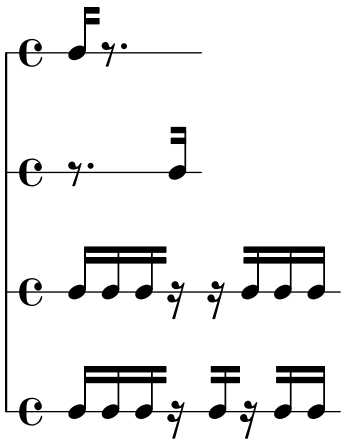
Perché i tratti di suddivisione alla fine di un gruppo di note unite da travatura siano rivolti a destra, si imposta `stemRightBeamCount` su un valore positivo. Perché i tratti di suddivisione all'inizio di un gruppo di note unite da travatura siano rivolti a sinistra, si imposta invece `stemLeftBeamCount` (Example 3).

Talvolta, ad esempio per una nota isolata circondata da pause, ha senso avere una coda che punti sia a destra che a sinistra. Lo si può fare con una coppia di indicatori di travatura `[]` da soli (Example 4).

(Nota che `\set stemLeftBeamCount` è sempre equivalente a `\once \set`. In altre parole, le impostazioni che definiscono il conteggio delle travature non "permangono", quindi la coppia di code attaccate al `c'16[]` solitario nell'ultimo esempio non hanno nulla a che fare con l'impostazione `\set` di due note prima.)

```
\score {
  <<
    % Example 1
    \new RhythmicStaff {
      \set stemLeftBeamCount = #0
      c16[]
      r8.
    }
    % Example 2
    \new RhythmicStaff {
      r8.
      \set stemRightBeamCount = #0
      c16[]
    }
    % Example 3
    \new RhythmicStaff {
      c16 c
      \set stemRightBeamCount = #2
      c16 r r
      \set stemLeftBeamCount = #2
      c16 c c
    }
    % Example 4
    \new RhythmicStaff {
      c16 c
      \set stemRightBeamCount = #2
      c16 r
      c16[]
      r16
      \set stemLeftBeamCount = #2
```

```
      c16 c
    }
  >>
}
```



Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 585, ⟨undefined⟩ [Grace notes], pagina ⟨undefined⟩.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Beam" in *Guida al Funzionamento Interno*, Sezione "BeamEvent" in *Guida al Funzionamento Interno*, Sezione "Beam_engraver" in *Guida al Funzionamento Interno*, Sezione "beam-interface" in *Guida al Funzionamento Interno*, Sezione "Stem_engraver" in *Guida al Funzionamento Interno*.

Travature a raggiera

Le travature a raggiera servono a indicare che un gruppo di note determinato deve essere eseguito a un tempo progressivamente accelerato (o rallentato), senza cambiare l'andamento complessivo del brano. L'estensione della travatura a raggiera deve essere indicato a mano con `[` e `]`, e la convergenza o divergenza delle travature si determina specificando la la direzione della proprietà `Beam` di `grow-direction`.
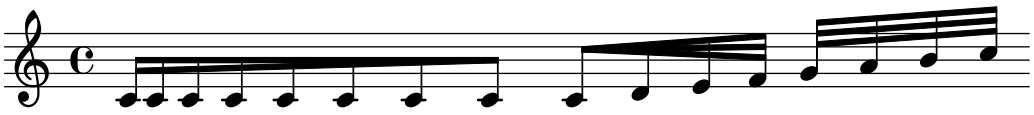
Perché il *ritardando* o l'*accelerando* indicati dalla travatura a raggiera trovino riscontro nella disposizione delle note e nell'esecuzione del file MIDI, le note devono essere raggruppate in un'espressione musicale delimitata da parentesi graffe e preceduta dal comando `featherDurations`, che specifica il rapporto tra le durate delle prime e delle ultime note del gruppo.

Le parentesi quadre indicano l'estensione della travatura, mentre quelle graffe indicano quali note devono avere una durata modificata. Di norma queste parentesi delimitano lo stesso gruppo di note, ma questo non è tassativo: i due comandi sono indipendenti.

Nell'esempio seguente le otto note da un sedicesimo occupano esattamente lo stesso tempo di una nota di due quarti, ma la prima nota dura la metà dell'ultima e le note intermedie si allungano gradualmente. Le prime quattro note da un trentaduesimo sono progressivamente più veloci, mentre le ultime quattro presentano lo stesso tempo.

```
\override Beam.grow-direction = #LEFT
\featherDurations #(ly:make-moment 2/1)
{ c16[ c c c c c c c] }
\override Beam.grow-direction = #RIGHT
```

```
\featherDurations #(ly:make-moment 2/3)
{ c32[ d e f] }
% ripristina le travature normali
\override Beam.grow-direction = #'()
{ g32[ a b c] }
```



La spaziatura rappresenta la durata effettiva delle note solo in modo approssimato, mentre il tempo nel file MIDI è esatto.

Comandi predefiniti

`\featherDurations`.

Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Problemi noti e avvertimenti

Il comando `\featherDurations` funziona solamente con frammenti di musica molto brevi e quando i numeri della frazione sono piccoli.
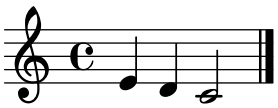
1.2.5 Battute

Stanghette

Le stanghette delimitano le misure e sono usate anche per indicare i ritornelli. Di norma, le stanghette semplici sono inserite automaticamente in base all’indicazione di tempo.

Si possono inserire altri tipi di stanghette col comando `\bar`. Ad esempio, di solito si usa una stanghetta finale al termine di un brano:

```
e4 d c2 \bar "|."
```



Se l’ultima nota di una misura non termina entro la stanghetta inserita automaticamente, non viene segnalato un errore: si presuppone che la nota continui nella misura successiva. Ma se ci sono tante misure simili in sequenza, la musica potrebbe apparire compressa oppure scorrere fuori dalla pagina. Questo accade perché le interruzioni di linea automatiche si verificano solo al termine di misure complete, ovvero quando tutte le note terminano prima dell’inizio di una misura.

> **Nota:** Una durata errata può impedire un’interruzione di linea, causando una linea di musica altamente compressa oppure a musica che prosegue fuori dalla pagina.

Le interruzioni di linea sono permesse anche in caso si stanghette inserite a mano anche all’interno di misure incomplete. Per permettere un’interruzione di linea senza che appaia una stanghetta si usa:

```
\bar ""
```
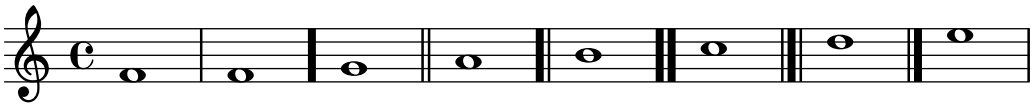
Questo comando inserirà una stanghetta invisibile e consentirà (senza però forzarla) un'interruzione di linea in questo punto. Il conteggio dei numeri di battuta non incrementa. Per forzare un'interruzione di linea si veda Sezione 4.3.1 [Line breaking], pagina 515.

Si possono inserire questa e altre stanghette speciali in qualsiasi punto. Quando coincidono con la fine di una misura, sostituiscono la stanghetta semplice che sarebbe stata posta automaticamente. Quando non coincidono con la fine di una misura, la stanghetta specificata viene inserita in quel punto.

Si noti che le stanghette manuali hanno una funzione puramente visiva. Non hanno alcun effetto sulle proprietà di una normale stanghetta, come i numeri della misura, le alterazioni, le interruzioni di linea, etc. Non influiscono nemmeno sul conteggio e sulla posizione delle stanghette automatiche successive. Quando una stanghetta manuale è posta nel punto in cui si trova già una normale stanghetta, le caratteristiche della stanghetta originale non sono alterate.

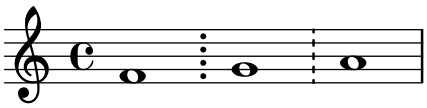
Sono disponibili per l'inserimento manuale due tipi di stanghette semplici e cinque tipi di stanghette doppie:

```
f1 \bar "|"
f1 \bar "."
g1 \bar "||"
a1 \bar ".|"
b1 \bar ".."
c1 \bar "|.|"
d1 \bar "|."
e1
```



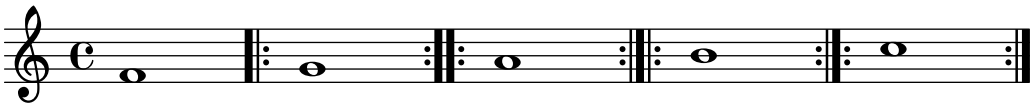
oltre alle stanghette puntate e tratteggiate:
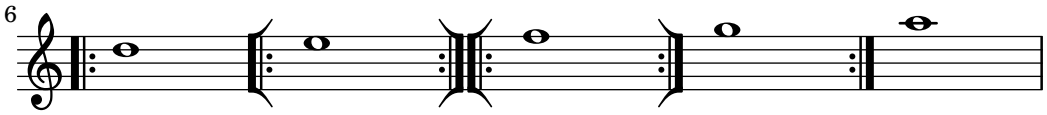
```
f1 \bar ";"
g1 \bar "!"
a1
```



e a nove tipi di stanghette per le ripetizioni:

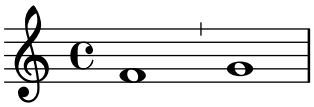
```
f1 \bar ".|:"
g1 \bar ":..:"
a1 \bar ":|.|:"
b1 \bar ":|.:"
c1 \bar ":.|.:"
d1 \bar "[|:"
e1 \bar ":|][|:"
f1 \bar ":|]"
g1 \bar ":|."
a1
```





Inoltre, una stanghetta può apparire come un semplice segno di spunta:

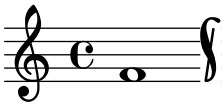
```
f1 \bar "'" g1
```



Tuttavia,dato che questi segni di spunta sono tipicamente usati nella notazione gregoriana, è preferibile usare `\divisioMinima`, come è descritto nella sezione [Divisiones], pagina 426 della parte dedicata al canto gregoriano.

Lilypond supporta la notazione gregoriana russa e fornisce una stanghetta speciale per questo tipo di notazione:

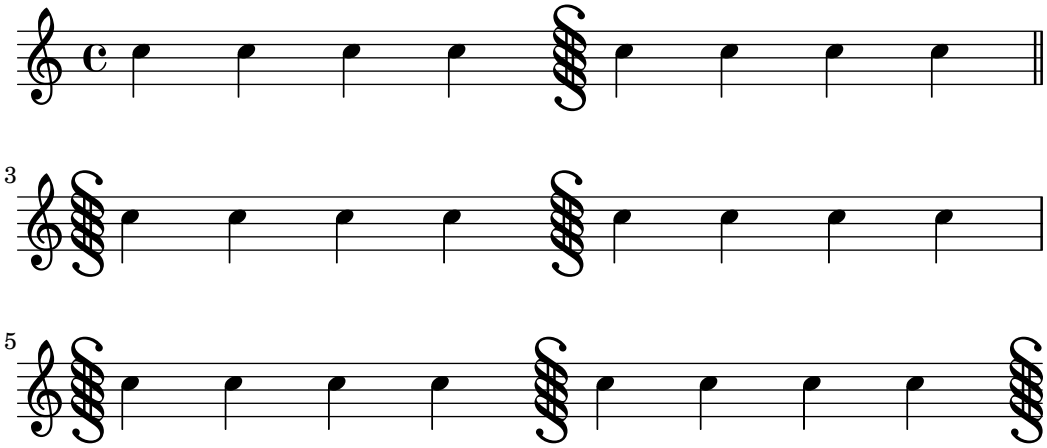
```
f1 \bar "k"
```

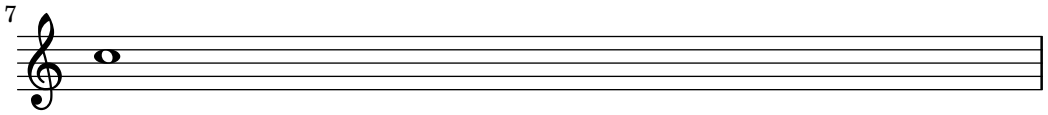


I dettagli di questo tipo di notazione sono spiegati in Sezione 2.9.5 [Typesetting Kievan square notation], pagina 435.

Per i segni di tipo segno interni al rigo, ci sono tre tipi di stanghette che differiscono nel comportamento quando incontrano un'interruzione di linea:

```
c4 c c c
\bar "S"
c4 c c c \break
\bar "S"
c4 c c c
\bar "S-|"
c4 c c c \break
\bar "S-|"
c4 c c c
\bar "S-S"
c4 c c c \break
\bar "S-S"
c1
```

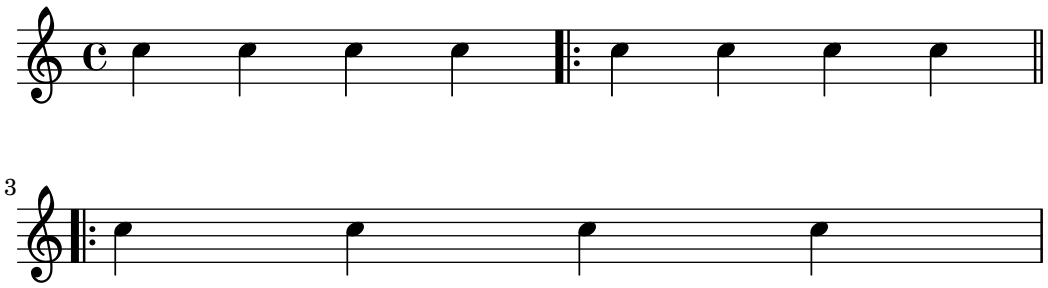




Sebbene LilyPond preveda l'inserimento manuale delle stanghette che indicano i ritornelli, ciò non consente il riconoscimento della musica come una sezione da ripetere. Tali sezioni devono essere inserite con i vari comandi di ripetizione (vedi ⟨undefined⟩ [Repeats], pagina ⟨undefined⟩), che creano automaticamente le stanghette appropriate.
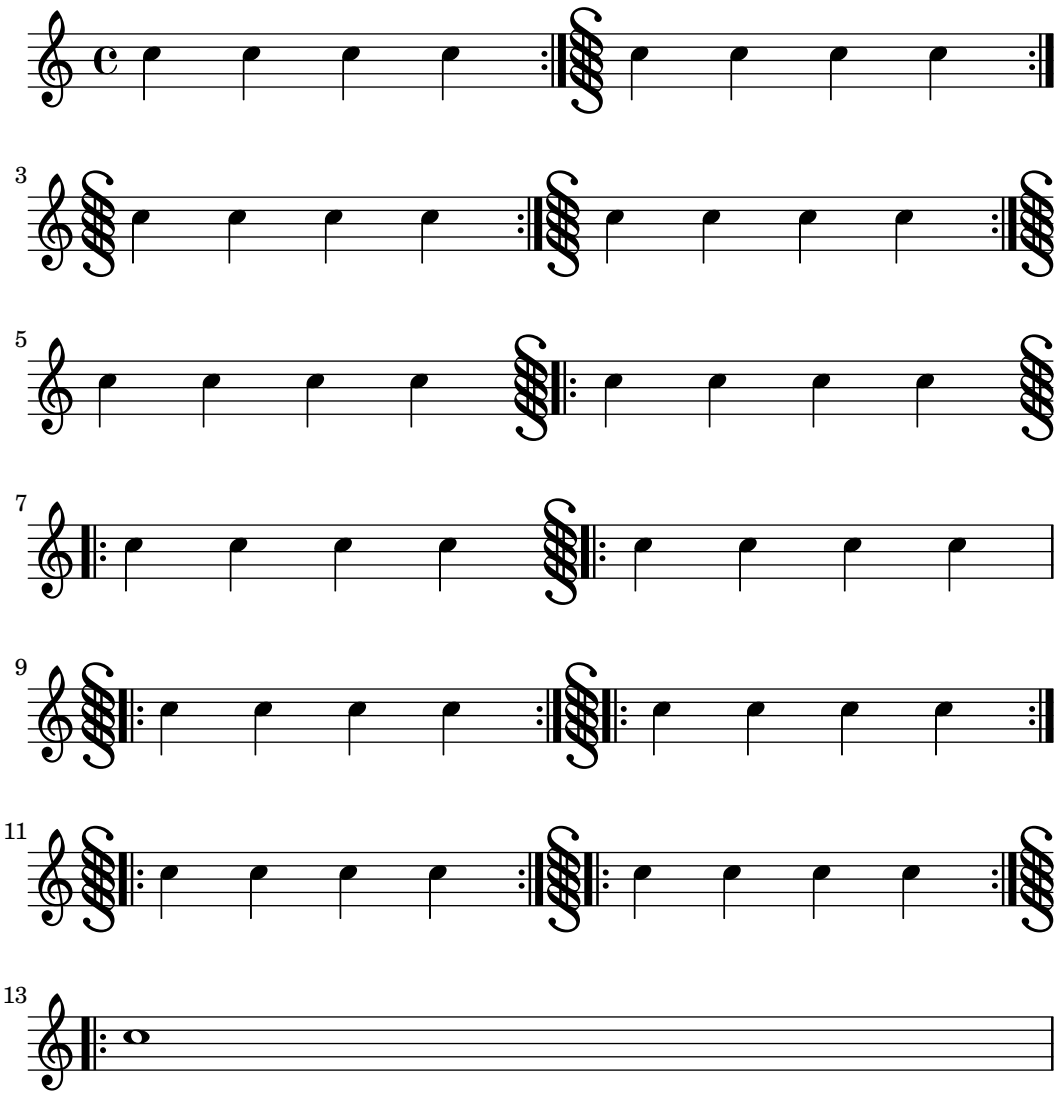
Inoltre si può specificare `".|:-||"`, che è equivalente a `".|:"` tranne in presenza di un'interruzione di linea, dove crea una doppia stanghetta alla fine della linea e una stanghetta di inizio ripetizione all'inizio della linea successiva.

```
c4 c c c
\bar ".|:-||"
c4 c c c \break
\bar ".|:-||"
c4 c c c
```



Esistono sei diverse combinazioni di ripetizioni e indicazioni di segno:

```
c4 c c c
\bar ":|.S"
c4 c c c \break
\bar ":|.S"
c4 c c c
\bar ":|.S-S"
c4 c c c \break
\bar ":|.S-S"
c4 c c c
\bar "S.|:-S"
c4 c c c \break
\bar "S.|:-S"
c4 c c c
\bar "S.|:"
c4 c c c \break
\bar "S.|:"
c4 c c c
\bar ":|.S.|:"
c4 c c c \break
\bar ":|.S.|:"
c4 c c c
\bar ":|.S.|:-S"
c4 c c c \break
\bar ":|.S.|:-S"
c1
```



Esiste inoltre un comando `\inStaffSegno` che crea una stanghetta con segno in congiunzione con un'appropriata stanghetta di ripetizione se usata con un comando `\repeat volta`, vedi ⟨undefined⟩ [Normal repeats], pagina ⟨undefined⟩..

Si possono definire nuovi tipi di stanghette con `\defineBarLine`:

`\defineBarLine` *`tipo-stanghetta #'(fine inizio span)`*

Le variabili di `\defineBarline` possono includere la stringa 'vuota' `""`,che è equivalente a una stanghetta invisibile. Oppure possono essere impostate su `#f`, che fa sì che non appaia alcuna stanghetta.
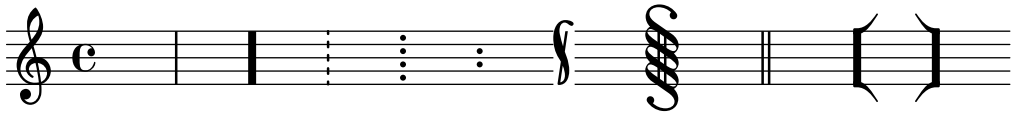
Dopo averla definita, si può richiamare la nuova stanghetta col comando `\bar` *tipo-stanghetta*.

Attualmente sono disponibile dieci tipi di stanghetta:

```
\defineBarLine ":" #'("" ":" "")
\defineBarLine "=" #'("=" "" "")
\defineBarLine "[" #'("" "[" "")
\defineBarLine "]" #'("]" "" "")

\new Staff {
  s1 \bar "|"
  s1 \bar "."
  s1 \bar "!"
  s1 \bar ";"
  s1 \bar ":"
```

```
  s1 \bar "k"
  s1 \bar "S"
  s1 \bar "="
  s1 \bar "["
  s1 \bar "]"
  s1 \bar ""
}
```

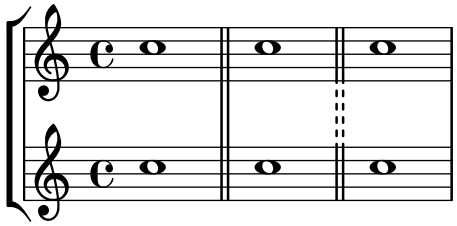


La stanghetta `"="` crea una stanghetta doppia da combinare con il il segno. Non va usata per creare una stanghetta doppia indipendente; in questo caso è preferibile usare `\bar "||"`.

Il segno `"-"` introduce le annotazioni alle stanghette che servono a distinguere quelle che hanno aspetto identico ma un diverso comportamento in corrispondenza delle interruzioni di linea e/o un diverso modo di connettere le stanghette tra i righi. La parte che segue il segno `"-"` non viene usato per costruire la stanghetta.

```
\defineBarLine "||-dashedSpan" #'("||" "" "!!")

\new StaffGroup <<
  \new Staff {
    c1 \bar "||"
    c1 \bar "||-dashedSpan"
    c1
  }
  \new Staff {
    c1
    c1
    c1
  }
>>
```

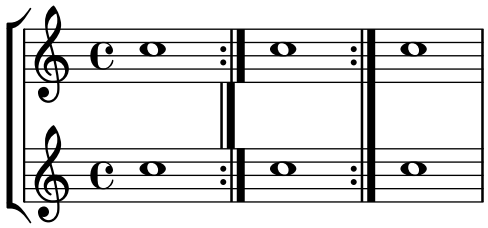


Inoltre, lo spazio `" "` fa da spaziatore e fa sì che le stanghette tra i righi siano allineate correttamente alle stanghette principali:

```
\defineBarLine ":|.-sbagliata" #'(":|." "" "|.")
\defineBarLine ":|.-giusta" #'(":|." "" " |.")

\new StaffGroup <<
  \new Staff {
    c1 \bar ":|.-sbagliata"
    c1 \bar ":|.-giusta"
    c1
  }
```

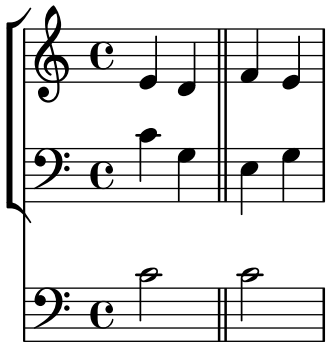
```
  \new Staff {
    c1
    c1
    c1
  }
>>
```



Se servono ulteriori elementi, LilyPond fornisce un modo semplice per definirli. Maggiori informazioni sulla modifica e l'aggiunta delle stanghette sono presenti nel file '`scm/bar-line.scm`'.

Nelle partiture con molti righi, un comando `\bar` inserito in un rigo viene applicato automaticamente a tutti i righi. Le stanghette risultanti sono connesse tra i diversi righi di un `StaffGroup`, `PianoStaff` o `GrandStaff`.

```
<<
  \new StaffGroup <<
    \new Staff {
      e4 d
      \bar "||"
      f4 e
    }
    \new Staff { \clef bass c4 g e g }
  >>
  \new Staff { \clef bass c2 c2 }
>>
```



Il comando '`\bar` *`tipo-stanghetta`*' è una scorciatoia di '`\set Timing.whichBar =` *`tipo-stanghetta`*'. Una stanghetta viene creata ogni volta che si imposta la proprietà `whichBar`.

Il tipo di stanghetta predefinita per le stanghette inserite automaticamente è `"|"`. Si può modificare in qualsiasi momento con '`\set Timing.defaultBarType =` *`tipo-stanghetta`*'.

Vedi anche

Guida alla notazione: Sezione 4.3.1 [Line breaking], pagina 515, ⟨undefined⟩ [Repeats], pagina ⟨undefined⟩, ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩.

File installati: '`scm/bar-line.scm`'.

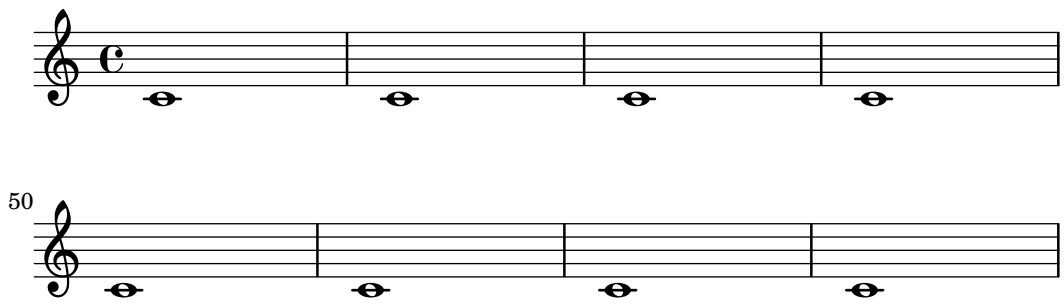
Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BarLine” in *Guida al Funzionamento Interno* (creata al livello `Staff`), Sezione “SpanBar” in *Guida al Funzionamento Interno* (tra i righi), Sezione “Timing_translator” in *Guida al Funzionamento Interno* (per le proprietà di Timing).

Numeri di battuta

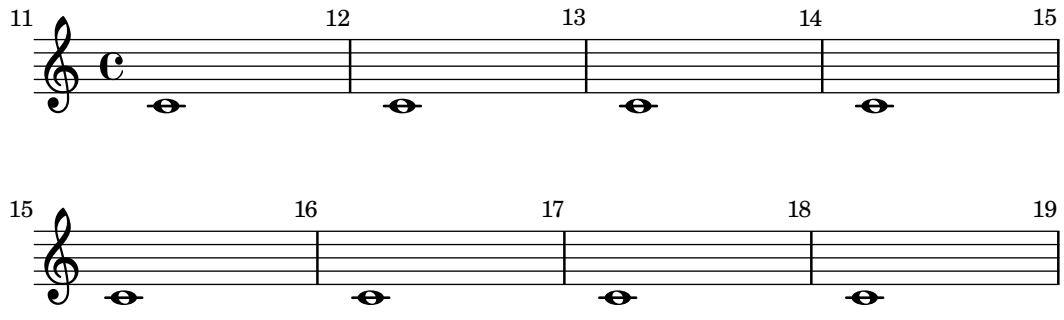
I numeri di battuta appaiono all’inizio di ogni linea tranne la prima. Il numero viene salvato nella proprietà `currentBarNumber`, che viene aggiornata automaticamente per ogni misura. Può anche essere impostata a mano:

```
c1 c c c
\break
\set Score.currentBarNumber = #50
c1 c c c
```



I numeri di battuta possono essere mostrati a intervalli regolari anziché solo all’inizio di ogni linea. Per farlo occorre sovrascrivere il comportamento predefinito e permettere ai numeri di battuta di apparire anche in punti diversi dall’inizio della linea. Questo comportamento è regolato dalla proprietà `break-visibility` di `BarNumber`, che considera tre valori impostabili su `#t` o `#f`, i quali indicano se il numero di battuta corrispondente debba essere visibile o no. L’ordine dei tre valori è `end of line visible`, `middle of line visible`, `beginning of line visible`. Nell’esempio seguente i numeri di battuta compaiono in tutti i punti possibili:

```
\override Score.BarNumber.break-visibility = ##(#t #t #t)
\set Score.currentBarNumber = #11
% Permette la visualizzazione del primo numero di battuta
\bar ""
c1 | c | c | c
\break
c1 | c | c | c
```

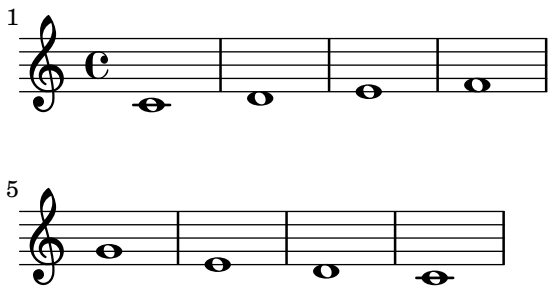


Frammenti di codice selezionati

Mostrare il numero di battuta nella prima misura

Il primo numero di battuta di una partitura viene soppresso se è inferiore o uguale a '1'. Se si imposta `barNumberVisibility` su `all-bar-numbers-visible`, verrà mostrato il numero di battuta della prima misura e di tutte quelle successive. Si noti che perché funzioni è necessario inserire una stanghetta invisibile prima della prima nota.

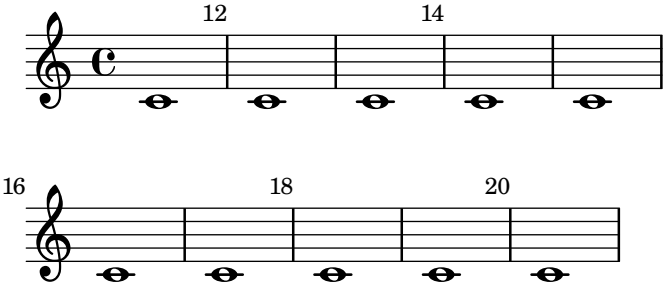
```
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Mostrare i numeri di battuta a intervalli regolari

I numeri di battuta possono essere resi visbili a intervalli regolari attraverso la proprietà `barNumberVisibility`. In questo esempio vengono mostrati ogni due misure eccetto alla fine della linea.

```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```



Numeri di battuta racchiusi in rettangoli o cerchi

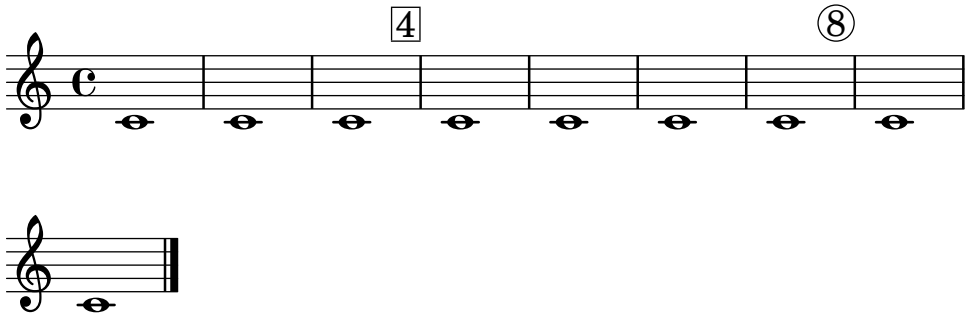
I numeri di battuta possono apparire anche all'interno di rettangoli o cerchi.

```
\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % Increase the size of the bar number by 2
  \override Score.BarNumber.font-size = #2

  % Draw a box round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
  \repeat unfold 5 { c1 }
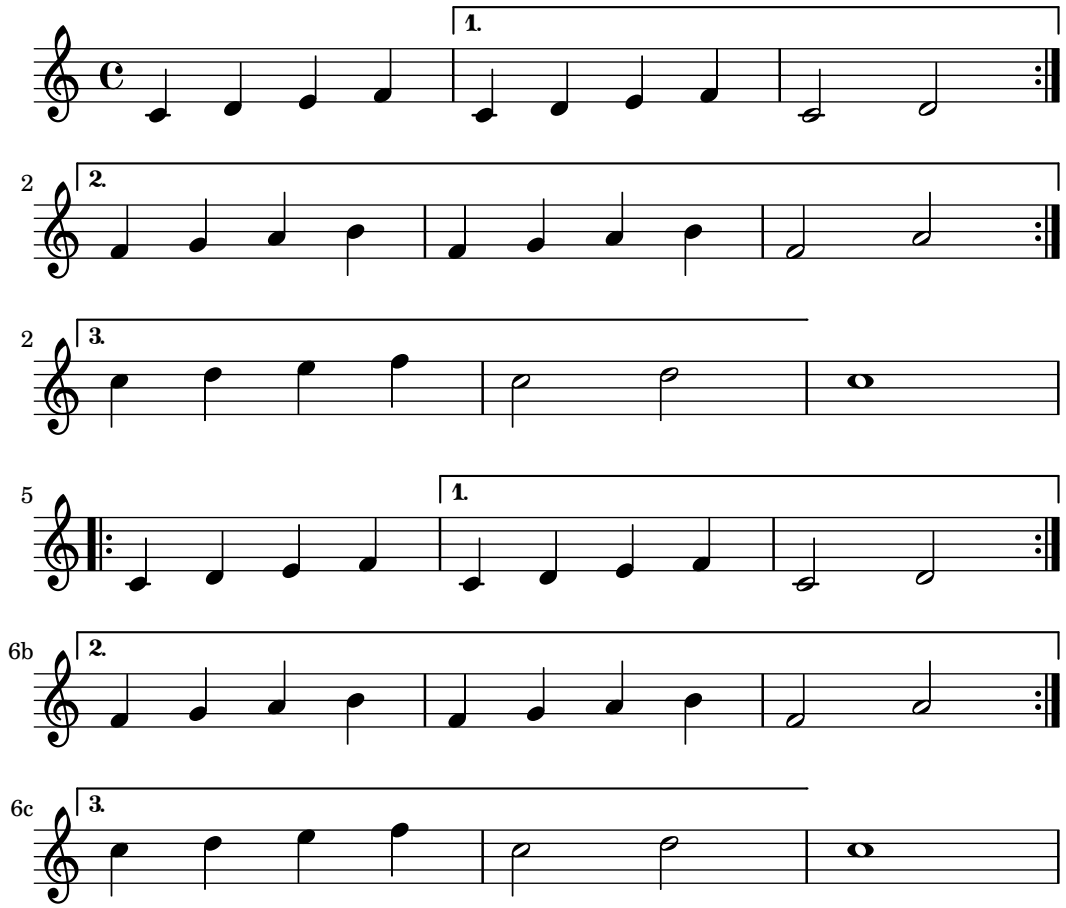
  % Draw a circle round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 4 { c1 } \bar "|."
}
```



Numeri di battuta alternativi

Si possono impostare due metodi alternativi di numerazione della battuta, utili specialmente per le ripetizioni.

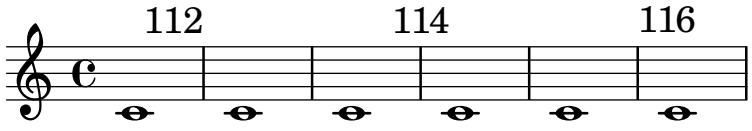
```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```



Allineare i numeri di battuta

Per impostazione predefinita i numeri di battuta sono allineati a destra rispetto al loro oggetto genitore. Di solito si tratta del margine sinistro della linea oppure, se i numeri appaiono all'interno della linea, del lato sinistro della stanghetta. I numeri possono essere posizionati anche direttamente sopra la stanghetta oppure allineati a sinistra della stanghetta.

```
\relative c' {
  \set Score.currentBarNumber = #111
  \override Score.BarNumber.break-visibility = #all-visible
  % Increase the size of the bar number by 2
  \override Score.BarNumber.font-size = #2
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c1
  % Center-align bar numbers
  \override Score.BarNumber.self-alignment-X = #CENTER
  c1 | c1
  % Left-align bar numbers
  \override Score.BarNumber.self-alignment-X = #LEFT
  c1 | c1
}
```
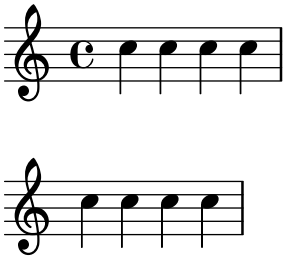


Togliere i numeri di battuta da uno spartito

I numeri di battuta possono essere tolti rimuovendo l'incisore `Bar_number_engraver` dal contesto `Score`.

```
\layout {
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}

\relative c'' {
  c4 c c c \break
  c4 c c c
}
```



Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BarNumber” in *Guida al Funzionamento Interno*, Sezione “Bar_number_engraver” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

I numeri di battuta possono collidere con la parte superiore della parentesi quadra di `StaffGroup`, se presente. Per evitare la collisione, si può usare la proprietà `padding` di `BarNumber` per posizionare correttamente il numero. Si veda Sezione “StaffGroup” in *Guida al Funzionamento Interno* e Sezione “BarNumber” in *Guida al Funzionamento Interno* per maggiori informazioni.

Controlli di battuta e del numero di battuta

I controlli di battuta aiutano a rilevare gli errori di durata. Il controllo di battuta si inserisce col simbolo della barra verticale, `|`, in un qualsiasi punto in cui è previsto l’inserimento di una stanghetta. Se vengono trovati controlli di battuta in punti diversi, viene creata una lista di avvisi nel file di log che mostra i numeri di linea e le linee in cui il controllo è fallito. Nell’esempio seguente il secondo controllo di battuta segnalerà un errore.

```
\time 3/4 c2 e4 | g2 |
```

I controlli di battuta possono essere usati anche all’interno del testo vocale:
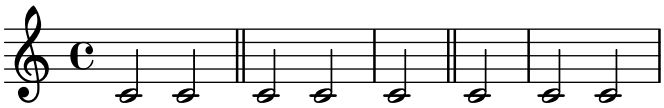
```
\lyricmode {
  \time 2/4
  Twin -- kle | Twin -- kle |
}
```

Una durata non corretta può generare uno spartito completamente alterato, specialmente nel caso di brani polifonici. Quindi il primo passo da compiere per correggere l’input è la verifica dei controlli di battuta e delle durate errate.

Se i controlli di battuta successivi sono spostati dello stesso intervallo musicale, viene mostrato solo il primo messaggio di avviso. Così l’avvertimento si concentra sulla causa dell’errore di tempo.

È anche possibile ridefinire l'azione da prendere quando si incontra un controllo di battuta o simbolo di barra verticale, `|`, nell'input, in modo che avvenga qualcosa di diverso dal controllo di battuta. Si può fare assegnando un'espressione musicale a `"|"`. Nell'esempio seguente `|`, invece di controllare la fine di una battuta, viene usato per inserire una stanghetta doppia ovunque appaia nell'input.

```
"|" = \bar "||"
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



Quando si copiano brani di una certa ampiezza, può essere d'aiuto verificare che i numeri di battuta di LilyPond corrispondano all'originale a partire dal quale si sta scrivendo il brano. Si può abilitare con `\barNumberCheck`, ad esempio,

```
\barNumberCheck #123
```

genererà un avvertimento se `currentBarNumber` non è 123 nel momento in cui viene elaborato.

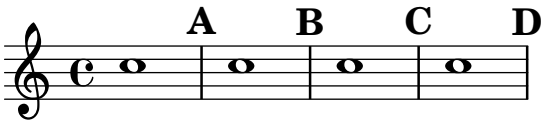
Vedi anche

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Segni di chiamata

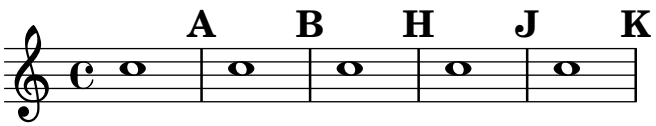
Per creare un segno di chiamata si usa il comando `\mark`.

```
c1 \mark \default
c1 \mark \default
c1 \mark \default
c1 \mark \default
```



Il segno viene incrementato automaticamente se si usa `\mark \default`, ma è possibile usare anche un numero intero come argomento in modo da impostare il segno manualmente. Il valore da usare viene salvato nella proprietà `rehearsalMark`.

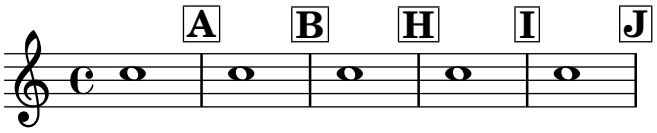
```
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



La lettera 'I' viene saltata, come vuole la tradizione tipografica. Se si desidera includere la lettera 'I', si può usare uno dei seguenti comandi, a seconda dello stile che si vuole (solo lettere, lettere in un quadrato o lettere in un cerchio).

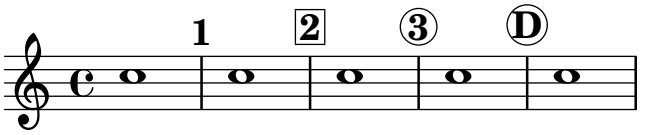
```
\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet
```

```
\set Score.markFormatter = #format-mark-box-alphabet
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



Lo stile viene definito dalla proprietà `markFormatter`. È una funzione che accoglie come argomenti il segno corrente (un numero intero) e il contesto corrente. Dovrebbe restituire un oggetto testuale. Nell'esempio seguente, `markFormatter` viene prima impostato su una procedura predefinita e dopo alcune misure su una procedura che produce un numero racchiuso in un quadrato.

```
\set Score.markFormatter = #format-mark-numbers
c1 \mark \default
c1 \mark \default
\set Score.markFormatter = #format-mark-box-numbers
c1 \mark \default
\set Score.markFormatter = #format-mark-circle-numbers
c1 \mark \default
\set Score.markFormatter = #format-mark-circle-letters
c1
```



Il file '`scm/translation-functions.scm`' contiene le definizioni di `format-mark-numbers` (il formato predefinito), `format-mark-box-numbers`, `format-mark-letters` e `format-mark-box-letters`. Possono essere usate come fonte di ispirazione per creare altre funzioni di formattazione.

Si possono usare `format-mark-barnumbers`, `format-mark-box-barnumbers` e `format-mark-circle-barnumbers` per ottenere i numeri di battuta invece di numeri o lettere crescenti.

Si possono specificare manualmente altri stili di segni di chiamata:

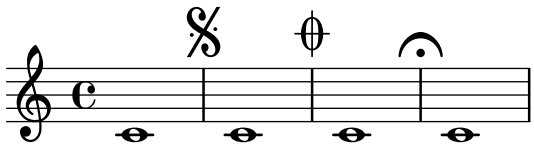
```
\mark "A1"
```

Si noti che `Score.markFormatter` non ha effetto sui segni specificati in questo modo. Tuttavia, è possibile applicare un `\markup` alla stringa.

```
\mark \markup{ \box A1 }
```

I glifi musicali (come il Segno) possono essere posti dentro il comando `\mark`

```
c1 \mark \markup { \musicglyph #"scripts.segno" }
c1 \mark \markup { \musicglyph #"scripts.coda" }
c1 \mark \markup { \musicglyph #"scripts.ufermata" }
c1
```



L'elenco dei simboli che possono essere prodotti con `\musicglyph` si trova in Sezione A.8 [The Feta font], pagina 632.

Per le più comuni modifiche relative al posizionamento dei segni di chiamata, si veda ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩. Per ottenere un controllo più preciso si consiglia di studiare il funzionamento della proprietà `break-alignable-interface` descritta in Sezione 5.5.1 [Aligning objects], pagina 600.

Il file '`scm/translation-functions.scm`' contiene le definizioni di `format-mark-numbers` e `format-mark-letters`, che possono essere usate come fonte di ispirazione per creare altre funzioni di formattazione.

Vedi anche

Guida alla notazione: Sezione A.8 [The Feta font], pagina 632, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, Sezione 5.5.1 [Aligning objects], pagina 600.

File installati: '`scm/translation-functions.scm`'.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.
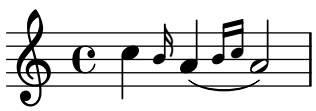
Guida al funzionamento interno: Sezione "MarkEvent" in *Guida al Funzionamento Interno*, Sezione "Mark_engraver" in *Guida al Funzionamento Interno*, Sezione "RehearsalMark" in *Guida al Funzionamento Interno*.

1.2.6 Questioni ritmiche particolari

Abbellimenti

Gli abbellimenti sono degli ornamenti musicali che hanno un carattere in corpo più piccolo e non alterano la durata della misura.
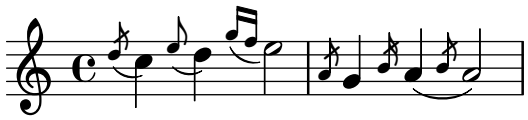
```
c4 \grace b16 a4(
\grace { b16 c16 } a2)
```



Esistono altri tre tipi di abbellimenti possibili; l'*acciaccatura* – un abbellimento in tempo libero indicato da una nota con legatura di portamento e un gambo barrato – e l'*appoggiatura*, che sottrae un valore determinato della nota principale a cui corrisponde e ha un gambo non barrato. È anche possibile creare un abbellimento con gambo barrato, come l'acciaccatura, ma privo di legatura di portamento, in modo da collocarla tra note già poste sotto una legatura: si usa il comando `\slashedGrace`.
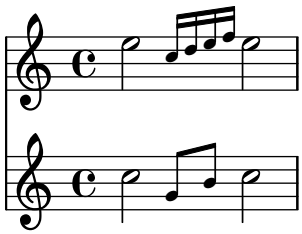
```
\acciaccatura d8 c4
\appoggiatura e8 d4
\acciaccatura { g16 f } e2
```

```
\slashedGrace a,8 g4
\slashedGrace b16 a4(
\slashedGrace b8 a2)
```
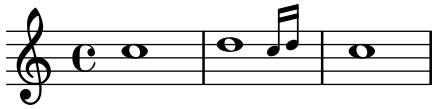


Il posizionamento degli abbellimenti è sincronizzato sui diversi righi. Nell'esempio seguente, ci sono due abbellimenti da un sedicesimo ogni abbellimento da un ottavo

```
<<
  \new Staff { e2 \grace { c16 d e f } e2 }
  \new Staff { c2 \grace { g8 b } c2 }
>>
```



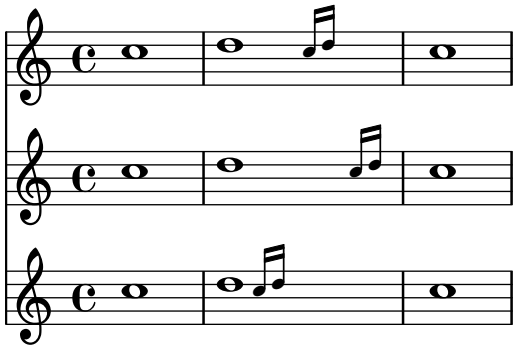
Se si desidera risolvere una nota su un abbellimento, si usa il comando `\afterGrace`. Considera due argomenti: la nota principale e gli abbellimenti che la seguono.

```
c1 \afterGrace d1 { c16[ d] } c1
```
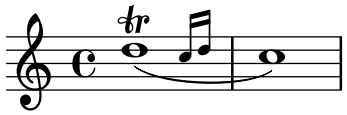


In questo modo, gli abbellimenti sono collocati dopo uno spazio corrispondente a 3/4 della durata della nota principale. La frazione predefinita 3/4 può essere modificata attraverso `afterGraceFraction`. L'esempio seguente mostra le diverse spaziature che si ottengono con la frazione predefinita, con 15/16 e infine con 1/2 della nota principale.

```
<<
  \new Staff {
    c1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff {
    #(define afterGraceFraction (cons 15 16))
    c1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff {
    #(define afterGraceFraction (cons 1 2))
    c1 \afterGrace d1 { c16[ d] } c1
  }
>>
```
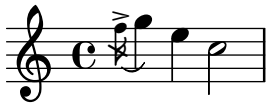


The space between the main note and the grace note may also be specified using spacers. The following example places the grace note after a space lasting 7/8 of the main note.

```
\new Voice {
  <<
    { d1^\trill_( }
    { s2 s4. \grace { c16 d } }
  >>
  c1)
}
```



L'espressione musicale introdotta dal comando `\grace` avrà delle impostazioni tipografiche speciali; per esempio, per rimpicciolire il tipo di carattere e impostare le direzioni. Dunque le modifiche che sovrascrivono tali impostazioni speciali devono essere poste all'interno del blocco `\grace`. Lo stesso vale per le modifiche che ripristinano i valori predefiniti. Nell'esempio seguente la direzione predefinita del gambo viene prima sovrascritta e poi ripristinata.

```
\new Voice {
  \acciaccatura {
    \stemDown
    f16->
    \stemNeutral
  }
  g4 e c2
}
```
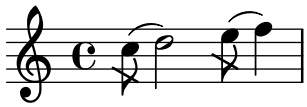


Frammenti di codice selezionati

Usare il gambo barrato degli abbellimenti con le teste normali

Il gambo barrato presente nelle acciaccature può essere applicato in altre situazioni.
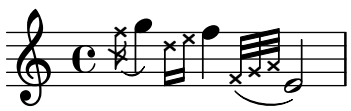
```
\relative c'' {
  \override Flag.stroke-style = #"grace"
  c8( d2) e8( f4)
}
```



Modificare l'aspetto degli abbellimenti di un intero brano

L'aspetto di tutte le espressioni contenute nei blocchi `\grace` di un brano può essere modificato con le funzioni `add-grace-property` e `remove-grace-property`. L'esempio seguente toglie la definizione della direzione di `Stem` nell'abbellimento, in modo che gli abbellimenti non siano sempre rivolti in su, e barra le teste di nota.

```
\relative c'' {
  \new Staff {
    $(remove-grace-property 'Voice 'Stem 'direction)
    $(add-grace-property 'Voice 'NoteHead 'style 'cross)
    \new Voice {
       \acciaccatura { f16 } g4
       \grace { d16 e } f4
       \appoggiatura { f,32 g a } e2
    }
  }
}
```
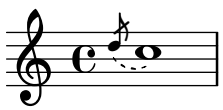


Ridefinire le impostazioni predefinite globali degli abbellimenti

Le impostazioni globali predefinite degli abbellimenti sono salvate negli identificatori `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic` e `stopAppoggiaturaMusic`, che sono definiti nel file `ly/grace-init.ly`. Ridefinendoli si possono ottenere effetti diversi.

```
startAcciaccaturaMusic = {
  <>(
  \override Flag.stroke-style = #"grace"
  \slurDashed
}

stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <>)
}

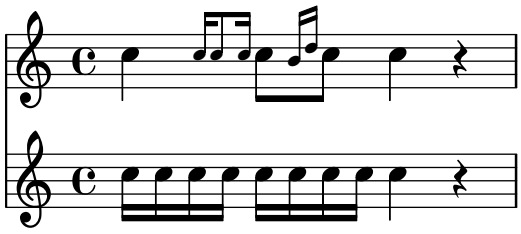
\relative c'' {
  \acciaccatura d8 c1
}
```



Posizionare gli abbellimenti con dello spazio fluttuante

Se si imposta la proprietà `'strict-grace-spacing`, le colonne musicali degli abbellimenti 'fluttuano', ovvero si scollegano dalle note normali: prima vengono spaziate le note normali, poi le colonne musicali degli abbellimenti vengono messe a sinistra delle colonne delle note principali.

```
\relative c'' {
  <<
    \override Score.SpacingSpanner.strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16 d } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}
```



Vedi anche

Glossario musicale: Sezione “acciaccatura” in *Glossario Musicale*, Sezione “acciaccatura” in *Glossario Musicale*, Sezione “appoggiatura” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Scaling durations], pagina ⟨undefined⟩, ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩.

File installati: ‘`ly/grace-init.ly`’.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.
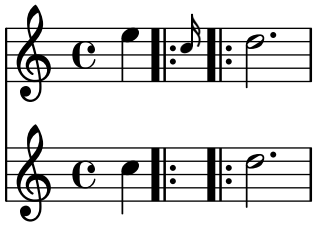
Guida al funzionamento interno: Sezione “GraceMusic” in *Guida al Funzionamento Interno*, Sezione “Grace_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace_auto_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace_spacing_engraver” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Una *acciaccatura* con molte note raggruppate sotto una travatura è priva della barra trasversale e ha il medesimo aspetto di una *appoggiatura* composta da varie note raggruppate sotto una travatura.
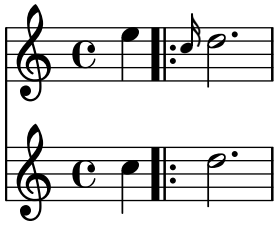
La sincronizzazione degli abbellimenti può nascondere delle sorprese, perché vengono sincronizzati anche altri elementi della notazione del rigo, come le armature di chiave, le stanghette, etc. Fai attenzione quando metti insieme righi che hanno degli abbellimenti con righi che non ne hanno. Ad esempio

```
<<
  \new Staff { e4 \bar ".|:" \grace c16 d2. }
  \new Staff { c4 \bar ".|:" d2. }
>>
```



Si può ovviare a questo problema inserendo degli abbellimenti della durata corrispondente negli altri righi. Riprendendo l'esempio precedente
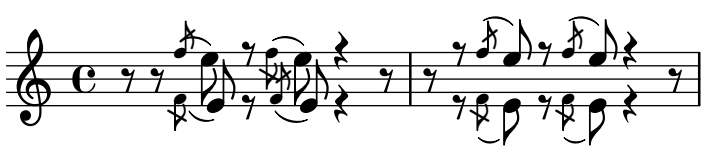
```
<<
  \new Staff { e4 \bar ".|:" \grace c16 d2. }
  \new Staff { c4 \bar ".|:" \grace s16 d2. }
>>
```



L'uso degli abbellimenti all'interno dei contesti della voce confonde il modo in cui la voce viene rappresentata. Si può evitare il problema inserendo una pausa o una nota tra il comando della voce e l'abbellimento.

```
accMusic = {
  \acciaccatura { f8 } e8 r8 \acciaccatura { f8 } e8 r4
}

\new Staff {
  <<
    \new Voice {
      \relative c'' {
        r8 r8 \voiceOne \accMusic \oneVoice r8 |
        r8 \voiceOne r8 \accMusic \oneVoice r8 |
      }
    }
    \new Voice {
      \relative c' {
        s8 s8 \voiceTwo \accMusic \oneVoice s8 |
        s8 \voiceTwo r8 \accMusic \oneVoice s8 |
      }
    }
  >>
}
```



Le sezioni con abbellimenti devono essere usate solamente all'interno di espressioni musicali sequenziali. Non è permesso annidare o affiancare gruppi di abbellimenti; potrebbero verificarsi blocchi del programma o altri errori se non si rispetta questa limitazione.

Ogni abbellimento generato nell'output MIDI ha una durata di 1/4 della sua vera durata. Se la durata complessiva degli abbellimenti è maggiore della durata della nota che li precede, verrà generato l'errore "`Going back in MIDI time`". A meno che non si diminuisca la durata degli abbellimenti, ad esempio:

```
c'8 \acciaccatura { c'8[ d' e' f' g'] }
```

diventa:

```
c'8 \acciaccatura { c'16[ d' e' f' g'] }
```

Oppure si cambia esplicitamente la durata musicale:

```
c'8 \acciaccatura { \scaleDurations 1/2 { c'8[ d' e' f' g'] } }
```

Vedi ⟨undefined⟩ [Scaling durations], pagina ⟨undefined⟩.

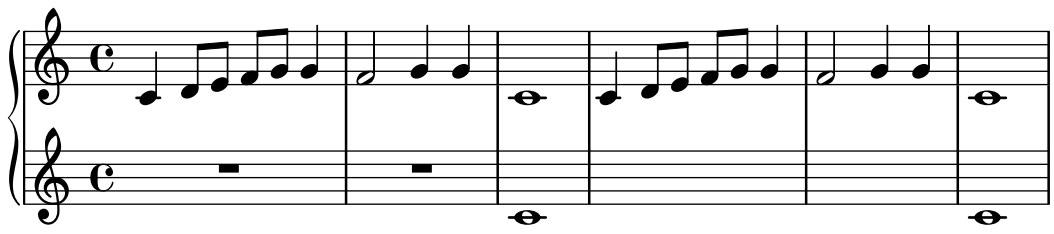
Allineamento sulle cadenze

Nell'ambito di una partitura per orchestra, le cadenze presentano un problema peculiare: quando si scrive una partitura che include una cadenza o un altro passaggio solistico, tutti gli altri strumenti devono saltare esattamente la durata complessiva delle note del passaggio, altrimenti inizieranno troppo presto o troppo tardi.

Una possibile soluzione a questo problema consiste nell'uso delle funzioni `mmrest-of-length` e `skip-of-length`. Queste funzioni Scheme prendono come argomento una sezione di musica salvata in una variabile e generano una pausa multipla o `\skip` della lunghezza esatta del brano.

```
MyCadenza = \relative c' {
  c4 d8 e f g g4
  f2 g4 g
}

\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(mmrest-of-length MyCadenza)
    c'1
    #(skip-of-length MyCadenza)
    c'1
  }
>>
```



Vedi anche

Glossario musicale: Sezione "cadenza" in *Glossario Musicale*.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Gestione del tempo

Il tempo è gestito da `Timing_translator`, che si trova nel contesto `Score`. Un suo alias, `Timing`, viene aggiunto nel contesto nel quale si trova `Timing_translator`. Per assicurarsi che l'alias `Timing` sia disponibile, occorre istanziare esplicitamente il contesto che lo contiene (come `Voice` o `Staff`).

Si usano le seguenti proprietà di `Timing` per tenere traccia del tempo in una partitura.

`currentBarNumber`
: Il numero di battuta corrente. Un esempio che mostra l'uso di questa proprietà si trova in ⟨undefined⟩ [Bar numbers], pagina ⟨undefined⟩.
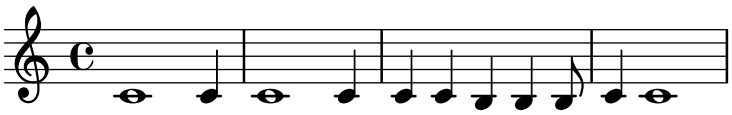
`measureLength`
: La durata delle misure nel tempo corrente. Per un tempo di 4/4 è 1, per un tempo di 6/8 è 3/4. Il suo valore determina quando debbano essere inserite le stanghette e come debbano essere generate le travature automatiche.

`measurePosition`
: Il punto della misura in cui ci si trova. Questa quantità viene reimpostata sottraendo `measureLength` ogni volta che `measureLength` viene raggiunto o superato. Quando questo accade, `currentBarNumber` viene incrementato.

`timing`
: Se impostato su `#t`, le variabili precedenti sono aggiornate ad ogni momento temporale. Se impostato su `#f`, l'incisore rimane nella misura corrente per un tempo indefinito.

Si può cambiare il tempo impostando esplicitamente una qualsiasi di queste variabili. Nel prossimo esempio, viene visualizzata l'indicazione di tempo predefinita di 4/4, ma `measureLength` è impostato su 5/4. A 4/8 della terza misura, `measurePosition` si sposta in avanti di 1/8 fino a 5/8, diminuendo quella misura di 1/8. Quindi la stanghetta successiva si troverà a 9/8 invece che a 5/4.

```
\new Voice \relative c' {
  \set Timing.measureLength = #(ly:make-moment 5/4)
  c1 c4 |
  c1 c4 |
  c4 c
  \set Timing.measurePosition = #(ly:make-moment 5/8)
  b4 b b8 |
  c4 c1 |
}
```



Come mostra l'esempio, `ly:make-moment n m` definisce una durata di n/m della nota intera. Ad esempio, `ly:make-moment 1 8` corrisponde alla durata di un ottavo mentre `ly:make-moment 7 16` a quella di sette sedicesimi.
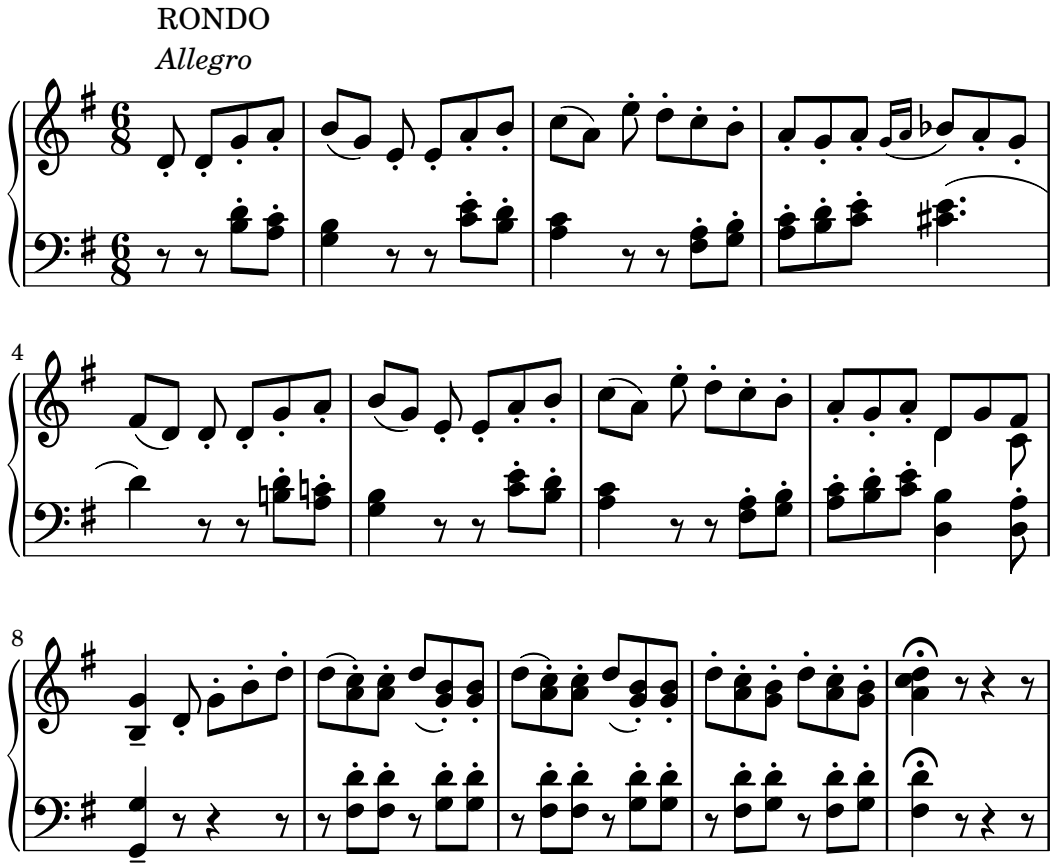
Vedi anche

Guida alla notazione: ⟨undefined⟩ [Bar numbers], pagina ⟨undefined⟩, ⟨undefined⟩ [Unmetered music], pagina ⟨undefined⟩.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Timing_translator” in *Guida al Funzionamento Interno*, Sezione “Score” in *Guida al Funzionamento Interno*.

1.3 Segni di espressione



Questa sezione elenca vari segni di espressione che si possono usare in una partitura.

1.3.1 Segni di espressione collegati alle note

Questa sezione spiega come creare segni di espressione collegati alle note: articolazioni, abbellimenti e dinamiche. Sono trattati anche i metodi per creare nuove indicazioni dinamiche.
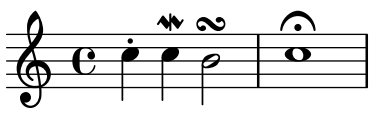
Articolazioni e abbellimenti

I diversi simboli che rappresentano articolazioni, ornamenti e altre indicazioni esecutive possono essere collegati a una nota con questa sintassi:

```
nota\nome
```
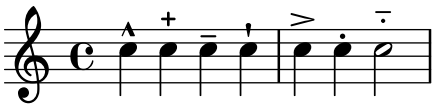
I valori possibili per *nome* sono elencati in Sezione A.13 [List of articulations], pagina 706. Ad esempio:

```
c4\staccato c\mordent b2\turn
c1\fermata
```



Alcune di queste articolazioni hanno delle abbreviazioni che ne semplificano l'inserimento. Le abbreviazioni sono attaccate al nome della nota e la loro sintassi è composta da un trattino - seguito da un simbolo che indica l'articolazione. Esistono abbreviazioni predefinite per *marcato*, *chiuso*, *tenuto*, *staccatissimo*, *accento*, *staccato* e *portato*. L'output corrispondente è:

```
c4-^ c-+ c-- c-!
c4-> c-. c2-_
```
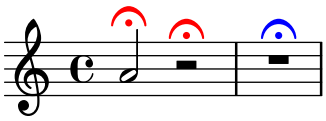


Le regole per il posizionamento predefinito delle articolazioni sono definite in 'scm/script.scm'. Articolazioni e ornamenti possono essere posizionati manualmente sopra o sotto il rigo; si veda Sezione 5.4.2 [Direction and placement], pagina 585.

Le articolazioni sono oggetti `Script`. Le loro proprietà sono descritte in dettaglio in Sezione "Script" in *Guida al Funzionamento Interno*.

Le articolazioni possono essere attaccate alle pause e alle note ma non alle pause multiple. Esiste un comando speciale predefinito, `\fermataMarkup`, che permette di attaccare un segno di corona a una pausa multipla (e soltanto ad essa). Questo crea un oggetto `MultiMeasureRestText`.

```
\override Script.color = #red
\override MultiMeasureRestText.color = #blue
a2\fermata r\fermata
R1\fermataMarkup
```



Oltre alle articolazioni, si può attaccare alle note anche un testo, posto tra virgolette o in un blocco `\markup{}`. Si veda ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩.

Ulteriori informazioni sull'ordine degli oggetti Script e TextScript collegati alle note si trovano in Sezione "Posizionamento degli oggetti" in *Manuale di Apprendimento*.

Frammenti di codice selezionati

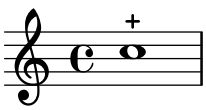
Modificare i valori predefiniti per le abbreviazioni delle articolazioni
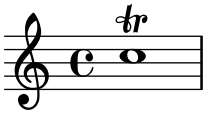
Le abbreviazioni sono definite in 'ly/script-init.ly', dove sono assegnati valori predefiniti alle variabili `dashHat`, `dashPlus`, `dashDash`, `dashBang`, `dashLarger`, `dashDot` e `dashUnderscore`. Questi valori predefiniti possono essere modificati. Ad esempio, per associare l'abbreviazione `-+` (`dashPlus`) al simbolo del trillo invece che al simbolo `+` predefinito, si assegna il valore `trill` alla variabile `dashPlus`:

```
\relative c'' { c1-+ }

dashPlus = "trill"

\relative c'' { c1-+ }
```
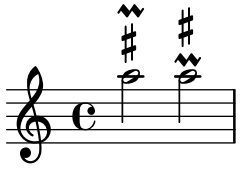




Controllo dell'ordine verticale degli script

L'ordine verticale degli script è determinato dalla proprietà `'script-priority`. Più il numero è piccolo, più sarà posto vicino alla nota. In questo esempio, il simbolo di diesis (oggetto `TextScript`) ha prima la priorità più bassa, dunque è posto più in basso nel primo esempio. Nel secondo, il trillo (oggetto `Script`) ha la priorità più bassa, quindi si trova all'interno. Quando due oggetti hanno la stessa priorità, l'ordine in cui sono inseriti determina quale viene prima.

```
\relative c''' {
  \once \override TextScript.script-priority = #-100
  a2^\prall^\markup { \sharp }
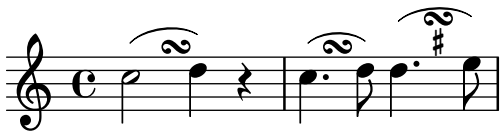
  \once \override Script.script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```



Creare un gruppetto ritardato

Creare un gruppetto ritardato, dove la nota più bassa del gruppetto usa l'alterazione, richiede vari \override. La proprietà `outside-staff-priority` deve essere impostata su `#f`, perché altrimenti questa avrebbe la precedenza sulla proprietà `avoid-slur`. Cambiando le frazioni `2/3` e `1/3` si aggiusta la posizione orizzontale.

```
\relative c'' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<
    { c4.( d8) }
    { s4 s\turn }
  >>
  \transpose c d \relative c'' <<
    { c4.( d8) }
    {
      s4
      \once \set suggestAccidentals = ##t
      \once \override AccidentalSuggestion #'outside-staff-priority = ##f
      \once \override AccidentalSuggestion #'avoid-slur = #'inside
      \once \override AccidentalSuggestion #'font-size = #-3
      \once \override AccidentalSuggestion #'script-priority = #-1
      \single \hideNotes
      b8-\turn \noBeam
      s8
    }
  >>
}
```



Vedi anche

Glossario Musicale: Sezione “tenuto” in *Glossario Musicale*, Sezione “accento” in *Glossario Musicale*, Sezione “staccato” in *Glossario Musicale*, Sezione “portato” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Posizionamento degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩, Sezione 5.4.2 [Direction and placement], pagina 585, Sezione A.13 [List of articulations], pagina 706, ⟨undefined⟩ [Trills], pagina ⟨undefined⟩.

File installati: ‘`scm/script.scm`’.

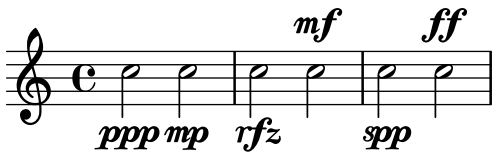
Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Script” in *Guida al Funzionamento Interno*, Sezione “TextScript” in *Guida al Funzionamento Interno*.

Dinamiche

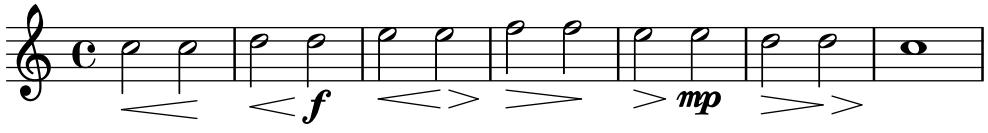
Le indicazioni dinamiche assolute si indicano con un comando che segue una nota, come ad esempio `c4\ff`. Le indicazioni dinamiche disponibili sono `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz` e `\rfz`. Le indicazioni dinamiche possono essere posizionate manualmente sopra o sotto il rigo, come è spiegato in dettaglio in Sezione 5.4.2 [Direction and placement], pagina 585.

```
c2\ppp c\mp
c2\rfz c^\mf
c2_\spp c^\ff
```



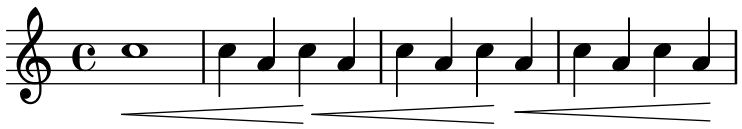
Un’indicazione di *crescendo* inizia con `\<` e termina con `\!`, un’indicazione dinamica assoluta o un’ulteriore indicazione di crescendo o decrescendo. Un’indicazione di *decrescendo* inizia con `\>` e termina nello stesso modo, ovvero con `\!`, un’indicazione dinamica assoluta oppure un altro segno di crescendo o decrescendo. Si possono usare `\cr` e `\decr` al posto di `\<` e `\>`. Le *forcelle* vengono create con questa notazione.

```
c2\< c\!
d2\< d\f
e2\< e\>
f2\> f\!
e2\> e\mp
d2\> d\>
c1\!
```
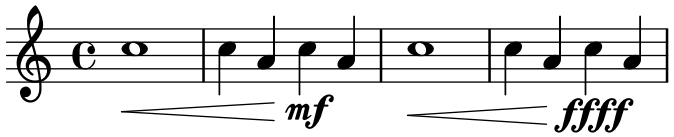


Una forcella che termina con `\!` si estenderà fino al margine destro della nota a cui è assegnato `\!`. Nel caso in cui sia terminata con l’inizio di un altro segno di *crescendo* o *decrescendo*, si estenderà fino al centro della nota a cui è assegnato il successivo `\<` o `\>`. La forcella successiva partirà dal margine destro della stessa nota invece che dal margine sinistro, come accade quando si termina con `\!`.

```
c1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
```



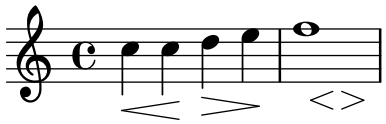
Le forcelle terminate con indicazioni dinamiche assolute invece che da `\!` avranno un aspetto simile. Tuttavia, la lunghezza dell'indicazione dinamica assoluta stessa può cambiare il punto in cui finisce la forcella precedente.

```
c1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
```
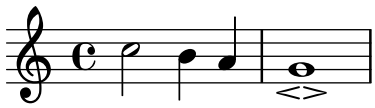


Occorre usare le pause spaziatrici per attaccare più di un'indicazione a una nota. Questo è utile soprattutto quando si aggiunge un *crescendo* e un *decrescendo* alla stessa nota:

```
c4\< c\! d\> e\!
<< f1 { s4 s4\< s4\> s4\! } >>
```
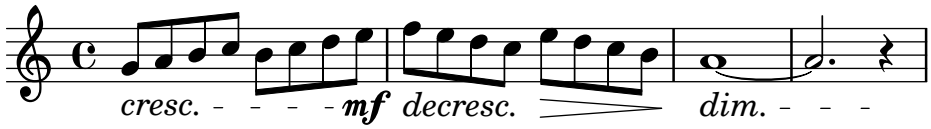


Il comando `\espressivo` permette di indicare un crescendo e un decrescendo sulla stessa nota. Tuttavia, si tenga presente che viene implementato come articolazione, non come dinamica.

```
c2 b4 a
g1\espressivo
```
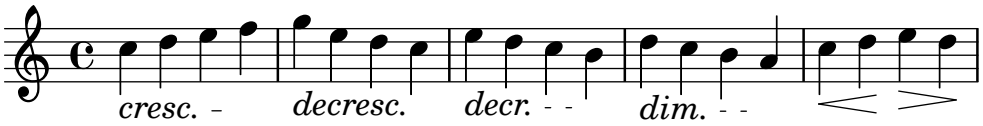


Le indicazioni di crescendo testuali iniziano con `\cresc`, quelle di decrescendo con `\decresc` o `\dim`. Le linee di estensione sono aggiunte automaticamente.

```
g8\cresc a b c b c d e\mf |
f8\decresc e d c e\> d c b |
a1\dim ~ |
a2. r4\! |
```



Le indicazioni testuali per i cambi di dinamica possono essere impiegate anche per sostituire le forcelle:

```
\crescTextCresc
c4\< d e f\! |
\dimTextDecresc
g4\> e d c\! |
\dimTextDecr
e4\> d c b\! |
\dimTextDim
d4\> c b a\! |
\crescHairpin
\dimHairpin
c4\< d\! e\> d\! |
```
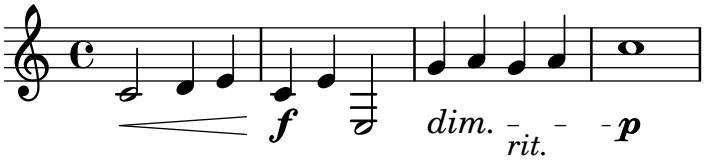


Per creare nuove indicazioni dinamiche assolute o testi da allineare alle dinamiche, si veda ⟨undefined⟩ [New dynamic marks], pagina ⟨undefined⟩.

Il posizionamento verticale della dinamica è gestito da Sezione "DynamicLineSpanner" in *Guida al Funzionamento Interno*.

Esiste un contesto `Dynamics` che permette di posizionare le indicazioni dinamiche su un'apposita linea orizzontale. Si usano le pause spaziatrici per indicarne la collocazione temporale (le note in un contesto `Dynamics` occupano infatti il rispettivo valore musicale, ma senza comparire sul rigo). Il contesto `Dynamics` può contenere altri elementi utili come indicazioni testuali, estensori del testo e indicazioni di pedalizzazione del pianoforte.

```
<<
  \new Staff \relative c' {
    c2 d4 e |
    c4 e e,2 |
    g'4 a g a |
    c1 |
  }
  \new Dynamics {
    s1\< |
    s1\f |
    s2\dim s2-"rit." |
    s1\p |
  }
>>
```



Comandi predefiniti

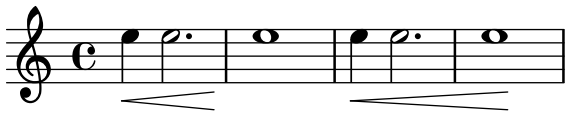
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

Frammenti di codice selezionati

Impostare il comportamento delle forcelle sulle stanghette

Se la nota che termina una forcella si trova sul primo battito di una battuta, la forcella si ferma prima della stanghetta che precede la nota. Si può controllare questo comportamento modificando la proprietà `'to-barline`.
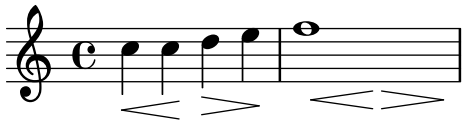
```
\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin.to-barline = ##f
  e4\< e2.
  e1\!
}
```



Impostare la lunghezza minima delle forcelle

Se le forcelle sono troppo corte, possono essere allungate modificando la proprietà `minimum-length` dell'oggetto `Hairpin`.
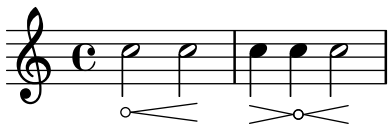
```
\relative c'' {
  c4\< c\! d\> e\!
  \override Hairpin.minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}
```



Forcelle con notazione al niente

Le forcelle di dinamica possono essere rappresentate con ua punta tonda (notazione "al niente") impostando la proprietà `circled-tip` dell'oggetto `Hairpin` su `#t`.

```
\relative c'' {
  \override Hairpin.circled-tip = ##t
  c2\< c\!
  c4\> c\< c2\!
}
```
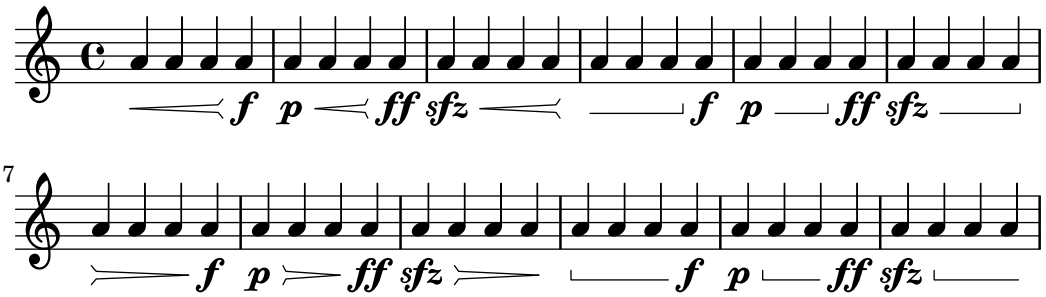


Stampare le forcelle in vari stili

Il segno di dinamica della forcella può avere diversi stili

```
\relative c'' {
  \override Hairpin.stencil = #flared-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
```

```
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
}
```



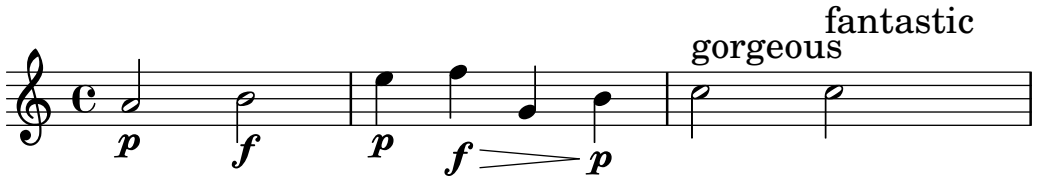
Dinamiche e segni testuali allineati verticalmente

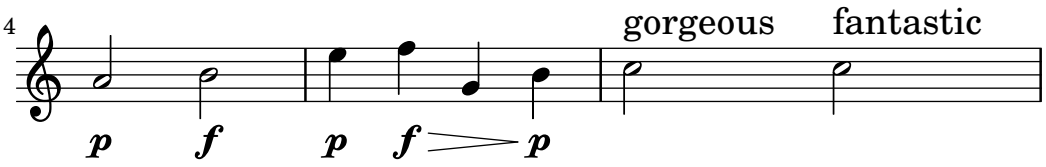
Tutti gli oggetti `DynamicLineSpanner` (forcelle e testi di dinamica) sono posti a una distanza minima dal rigo determinata da `'staff-padding`. Se si imposta `'staff-padding` su un valore abbastanza grande, le dinamiche saranno allineate.

```
\markup \vspace #1 %avoid LSR-bug

music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = #3
  \textLengthOn
  \override TextScript.staff-padding = #1
  \music
}
```

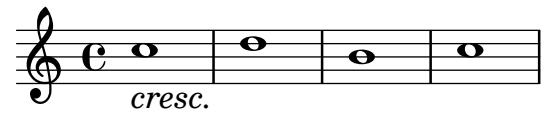




Nascondere la linea di estensione per le dinamiche testuali

I cambi di dinamica in stile testuale (come cresc. e dim.) appaiono con una linea tratteggiata che mostra la loro estensione. Questa linea può essere soppressa nel modo seguente:

```
\relative c'' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```
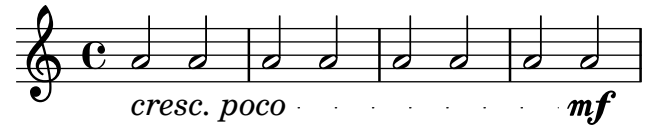


Nascondere la linea di estensione per le dinamiche testuali

Il testo usato per i crescendo e i decrescendo può essere cambiato modificando le proprietà di contesto `crescendoText` e `decrescendoText`.

Lo stile della linea dell'estensore può essere cambiato modificando la proprietà `'style` di `DynamicTextSpanner`. Il valore predefinito è `'dashed-line`; gli altri valori possibili sono `'line`, `'dotted-line` e `'none`.

```
\relative c'' {
  \set crescendoText = \markup { \italic { cresc. poco } }
  \set crescendoSpanner = #'text
  \override DynamicTextSpanner.style = #'dotted-line
  a2\< a
  a2 a
  a2 a
  a2 a\mf
}
```



Vedi anche

Glossario Musicale: Sezione "al niente" in *Glossario Musicale*, Sezione "crescendo" in *Glossario Musicale*, Sezione "decrescendo" in *Glossario Musicale*, Sezione "forcella" in *Glossario Musicale*. Manuale di apprendimento: Sezione "Articolazione e dinamiche" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 585, ⟨undefined⟩ [New dynamic marks], pagina ⟨undefined⟩, Sezione 3.5.3 [What goes into the MIDI output?], pagina 494, Sezione 3.5.5 [Controlling MIDI dynamics], pagina 495.

Frammenti: Sezione "Expressive marks" in *Frammenti di codice*.
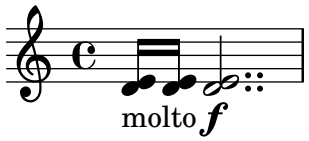
Guida al funzionamento interno: Sezione "DynamicText" in *Guida al Funzionamento Interno*, Sezione "Hairpin" in *Guida al Funzionamento Interno*, Sezione "DynamicLineSpanner" in *Guida al Funzionamento Interno*, Sezione "Dynamics" in *Guida al Funzionamento Interno*.

Nuove indicazioni dinamiche

Il modo più semplice per creare indicazioni dinamiche è usare gli oggetti `\markup`.

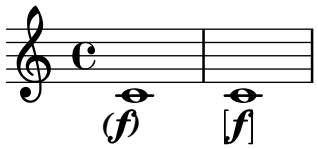
```
moltoF = \markup { molto \dynamic f }

\relative c' {
  <d e>16_\moltoF <d e>
  <d e>2..
}
```
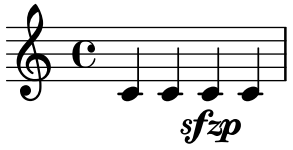


In modalità markup, si possono creare dinamiche editoriali (racchiuse tra parentesi normali o quadrate). La sintassi della modalità markup è descritta in ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

```
roundF = \markup {
    \center-align \concat { \bold { \italic ( }
           \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative c' {
  c1_\roundF
  c1_\boxF
}
```



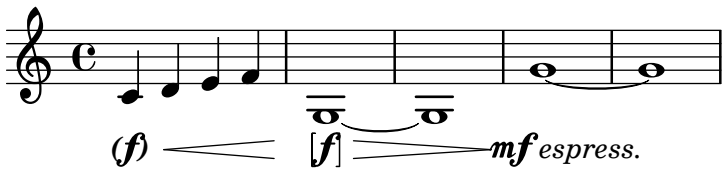
È possibile creare semplicemente indicazioni dinamiche centrate verticalmente con la funzione `make-dynamic-script`.

```
sfzp = #(make-dynamic-script "sfzp")
\relative c' {
  c4 c c\sfzp c
}
```



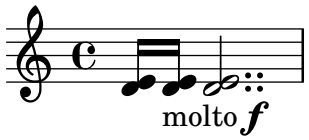
In generale, `make-dynamic-script` assume come argomento qualsiasi oggetto markup. Il tipo di carattere per la dinamica contiene solo i caratteri `f,m,p,r,s` e `z`; dunque, se si desidera creare un'indicazione dinamica che contenga testo semplice e simboli di punteggiatura, occorre usare dei comandi markup che ripristinino la famiglia e la codifica del tipo di carattere per il testo normale, ad esempio `\normal-text`. Il vantaggio nell'uso di `make-dynamic-script` al posto di un normale markup è l'allineamento verticale degli oggetti markup e delle forcelle collegate alla stessa testa di nota.

```
roundF = \markup { \center-align \concat {
           \normal-text { \bold { \italic ( } }
           \dynamic f
           \normal-text { \bold { \italic ) } } } }
boxF = \markup { \bracket { \dynamic f } }
mfEspress = \markup { \center-align \line {
              \hspace #3.7 mf \normal-text \italic espress. } }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative c' {
  c4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g1
  g'1~\mfEspressDynamic
  g1
}
```
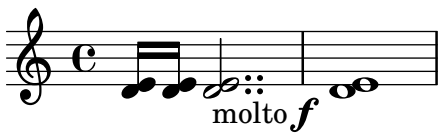


Si può usare anche la forma Scheme della modalità markup. La sintassi è spiegata in Sezione "Markup construction in Scheme" in *Estendere*.

```
moltoF = #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative c' {
  <d e>16 <d e>
  <d e>2..\moltoF
}
```



Per allineare a sinistra il testo di dinamica invece di centrarlo su una nota, si usa un `\tweak`:

```
moltoF = \tweak DynamicText.self-alignment-X #LEFT
         #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative c' {
  <d e>16 <d e>
  <d e>2..\moltoF <d e>1
}
```



Le impostazioni dei tipi di carattere in modalità markup sono descritti in ⟨undefined⟩ [Selecting font and font size], pagina ⟨undefined⟩.

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, ⟨undefined⟩ [Selecting font and font size], pagina ⟨undefined⟩, Sezione 3.5.3 [What goes into the MIDI output?], pagina 494, Sezione 3.5.5 [Controlling MIDI dynamics], pagina 495.

Extending LilyPond: Sezione “Markup construction in Scheme” in *Estendere*.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

1.3.2 Indicazioni espressive curvilinee

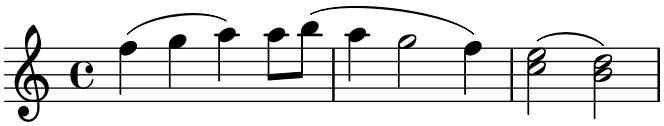
Questa sezione spiega come creare varie indicazioni espressive con forma curvilinea: legature di portamento, legature di frase, respiri, portamenti indeterminati discendenti (cadute) o ascendenti.

Legature di portamento

Le *legature di portamento* si inseriscono con delle parentesi:

> **Nota:** Nella musica polifonica, una legatura di portamento deve terminare nella stessa voce in cui è iniziata.

```
f4( g a) a8 b(
a4 g2 f4)
<c e>2( <b d>2)
```
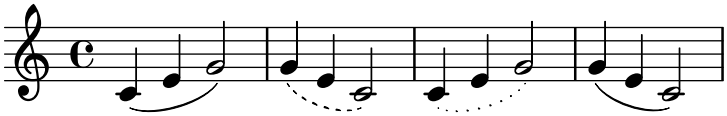


Le legature di portamento possono essere posizionate manualmente sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 585.

Non è possibile tracciare due legature di portamento simultanee o sovrapposte, ma si può ricorrere a una legatura di frase posta per mostrare più legature. Questo permette di creare contemporaneamente due legature di portamento. Maggiori dettagli si trovano in ⟨undefined⟩ [Phrasing slurs], pagina ⟨undefined⟩.
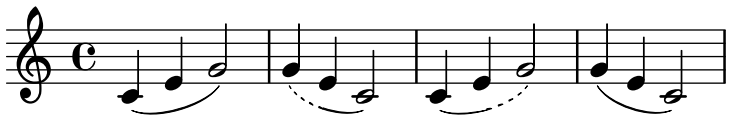
Le legature di portamento possono essere continue, punteggiate o tratteggiate. Lo stile predefinito è quello continuo:

```
c4( e g2)
\slurDashed
g4( e c2)
\slurDotted
c4( e g2)
\slurSolid
g4( e c2)
```
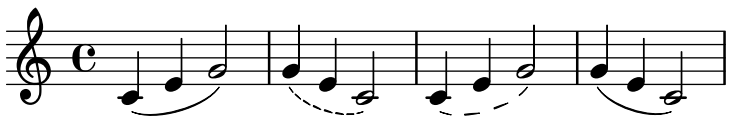


Le legature di portamento possono essere anche semitratteggiate (half-dashed), ovvero con la prima metà tratteggiata e la seconda continua; oppure semicontinue (half-solid), ovvero con la prima metà continua e la seconda tratteggiata:

```
c4( e g2)
\slurHalfDashed
g4( e c2)
\slurHalfSolid
c4( e g2)
\slurSolid
g4( e c2)
```



Si possono definire modelli di tratteggio personalizzati per le legature di portamento:

```
c4( e g2)
\slurDashPattern #0.7 #0.75
g4( e c2)
\slurDashPattern #0.5 #2.0
c4( e g2)
\slurSolid
g4( e c2)
```



Comandi predefiniti
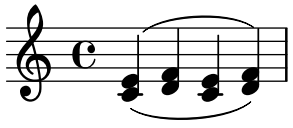
`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`, `\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

Frammenti di codice selezionati

Uso delle doppie legature di portamento per gli accordi legati

Alcuni compositori scrivono due legature di portamento per indicare gli accordi legati. Si può ottenere questo risultato impostando `doubleSlurs`.
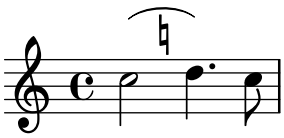
```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```



Posizionare il testo a margine dentro le legature di portamento

I testi a margine devono avere la proprietà `outside-staff-priority` impostata su false per poter apparire dentro le legature di portamento.

```
\relative c'' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```
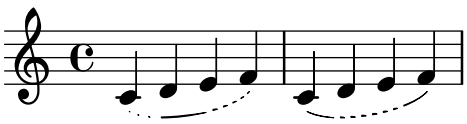


Legature di portamento con complesse strutture di tratteggio

Le legature di portamento possono avere schemi di tratteggio complessi definendo la proprietà `dash-definition`. `dash-definition` è una lista di `dash-elements`. Un `dash-element` è una lista di parametri che definiscono il comportamento del tratteggio per un segmento della legatura.

La legatura di portamento è definita come il parametro t della curva di bezier che va da 0 sul margine sinistro della legatura fino a 1 su quello destro. `dash-element` è una lista di `(inizio-t fine-t frazione-trattino punto-trattino)`. La regione della legatura di portamento che va da `inizio-t` a `fine-t` avrà una frazione `frazione-trattino` di ogni `punto-trattino` nero. `punto-trattino` viene definito in spazi rigo. `frazione-trattino` è impostato su 1 per una legatura di portamento continua.

```
\relative c' {
  \once \override
    Slur.dash-definition = #'((0 0.3 0.1 0.75)
                              (0.3 0.6 1 1)
                              (0.65 1.0 0.4 0.75))
  c4( d e f)
  \once \override
    Slur.dash-definition = #'((0 0.25 1 1)
                              (0.3 0.7 0.4 0.75)
                              (0.75 1.0 1 1))
  c4( d e f)
}
```



Vedi anche

Glossario Musicale: Sezione “legatura di portamento” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Sul non annidamento di parentesi e legature di valore” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 585, ⟨undefined⟩ [Phrasing slurs], pagina ⟨undefined⟩.
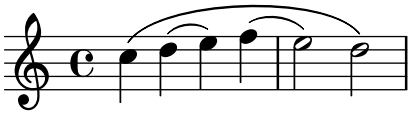
Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Slur” in *Guida al Funzionamento Interno*.

Legature di frase

Le *legature di frase*, che indicano una frase musicale, si scrivono con i comandi `\(` e `\)`:

```
c4\( d( e) f(
e2) d\)
```
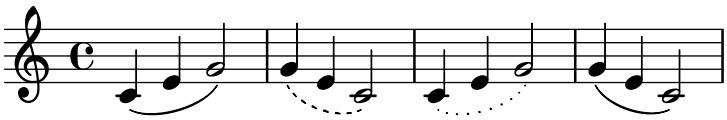


A livello tipografico, una legatura di frase si comporta in modo pressoché identico a una normale legatura di portamento. Sono però trattate come oggetti diversi; ad esempio, `\slurUp` non ha effetto su una legatura di frase. Le legature di frase possono essere posizionate sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 585.

Più legature di frase simultanee o sovrapposte non sono permesse.
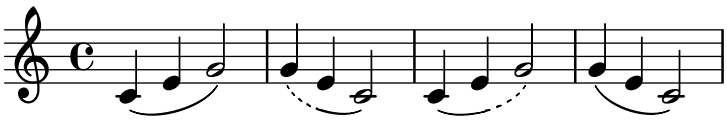
Le legature di frase possono essere continue, puntate o tratteggiate. Lo stile predefinito è quello continuo:

```
c4\( e g2\)
\phrasingSlurDashed
g4\( e c2\)
\phrasingSlurDotted
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```



Le legature di frase possono essere anche semitratteggiate (la prima metà tratteggiata, la seconda continua) o semicontinue (la prima metà continua, la seconda tratteggiata):
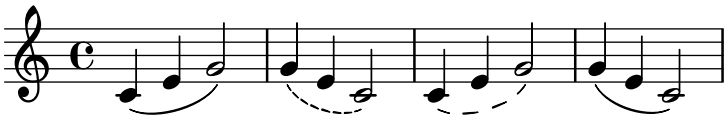
```
c4\( e g2\)
\phrasingSlurHalfDashed
g4\( e c2\)
\phrasingSlurHalfSolid
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```



Si possono definire modelli di tratteggio personalizzati anche per le legature di frase:

```
c4\( e g2\)
\phrasingSlurDashPattern #0.7 #0.75
g4\( e c2\)
\phrasingSlurDashPattern #0.5 #2.0
c4\( e g2\)
\phrasingSlurSolid
```

```
g4\( e c2\)
```



Le definizioni dei modelli di tratteggio per le legature di frase hanno la stessa struttura di quelle per le legature di portamento. Per maggiori informazioni sui modelli complessi di tratteggio si consultino i frammenti in ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩.

Comandi predefiniti

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`, `\phrasingSlurDashed`, `\phrasingSlurDotted`, `\phrasingSlurHalfDashed`, `\phrasingSlurHalfSolid`, `\phrasingSlurDashPattern`, `\phrasingSlurSolid`.

Vedi anche

Manuale di apprendimento: Sezione “Sul non annidamento di parentesi e legature di valore” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 585, ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩.
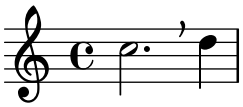
Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “PhrasingSlur” in *Guida al Funzionamento Interno*.

Respiri
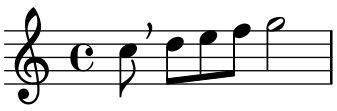
I respiri si inseriscono col comando `\breathe`:

```
c2. \breathe d4
```



Un respiro termina una travatura automatica; per evitare questo comportamento, si veda ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩.

```
c8 \breathe d e f g2
```



È supportata la divisio, indicatore del respiro nella musica antica. Maggiori dettagli in [Divisiones], pagina 426.
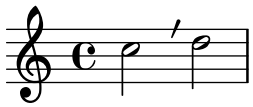
Frammenti di codice selezionati

Cambiare il simbolo del segno di respiro

Il glifo del respiro può essere modificato sovrascrivendo la proprietà `text` dell’oggetto di formattazione `BreathingSign` con qualsiasi testo incluso in un blocco markup.

```
\relative c'' {
  c2
  \override BreathingSign.text =
    \markup { \musicglyph #"scripts.rvarcomma" }
```
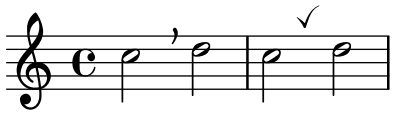
```
  \breathe
  d2
}
```



Usare un segno di spunta come simbolo di respiro

La musica vocale e per fiati usa frequentemente il segno di spunta come segno di respiro. Questo indica un respiro che sottrae un po' di tempo alla nota precedente invece di prendere una piccola pausa, indicata dal segno di respiro rappresentato dalla virgola. Il segno può essere spostato un po' su per allontanarlo dal rigo.

```
\relative c'' {
  c2
  \breathe
  d2
  \override BreathingSign.Y-offset = #2.6
  \override BreathingSign.text =
    \markup { \musicglyph #"scripts.tickmark" }
  c2
  \breathe
  d2
}
```
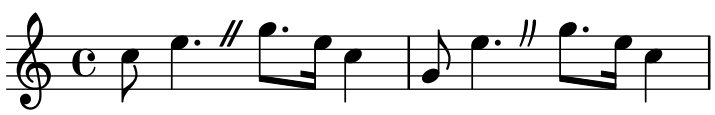


Inserire una cesura

I segni di cesura possono essere creati sovrascrivendo la proprietà `'text` dell'oggetto `BreathingSign`. È disponibile anche un segno di cesura curvo.

```
\relative c'' {
  \override BreathingSign.text = \markup {
    \musicglyph #"scripts.caesura.straight"
  }
  c8 e4. \breathe g8. e16 c4

  \override BreathingSign.text = \markup {
    \musicglyph #"scripts.caesura.curved"
  }
  g8 e'4. \breathe g8. e16 c4
}
```



Vedi anche

Glossario Musicale: Sezione “cesura” in *Glossario Musicale*.

Guida alla notazione: [Divisiones], pagina 426.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BreathingEvent” in *Guida al Funzionamento Interno*, Sezione “BreathingSign” in *Guida al Funzionamento Interno*, Sezione “Breathing_sign_engraver” in *Guida al Funzionamento Interno*.

Portamenti indeterminati discendenti (cadute) e ascendenti

I *portamenti indeterminati verso il basso (cadute) e verso l’alto* possono essere aggiunti alle note col comando `\bendAfter`. La direzione del portamento è indicata con un più o un meno (su o giù). Il numero indica l’intervallo per cui il portamento si estenderà *oltre* la nota principale.

```
c2\bendAfter #+4
c2\bendAfter #-4
c2\bendAfter #+6.5
c2\bendAfter #-6.5
c2\bendAfter #+8
c2\bendAfter #-8
```



Frammenti di codice selezionati

Cambiare la forma dei portamenti indeterminati verso il basso o verso l’alto

La proprietà `shortest-duration-space` può essere modificata per cambiare la forma dei portamenti indeterminati verso il basso o verso l’alto.

```
\relative c'' {
  \override Score.SpacingSpanner.shortest-duration-space = #4.0
  c2-\bendAfter #5
  c2-\bendAfter #-4.75
  c2-\bendAfter #8.5
  c2-\bendAfter #-6
}
```



Vedi anche

Glossario Musicale: Sezione “portamento indeterminato verso il basso” in *Glossario Musicale*, Sezione “portamento indeterminato verso l’alto” in *Glossario Musicale*.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

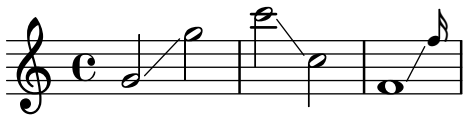
1.3.3 Indicazioni espressive lineari

Questa sezione spiega come creare varie indicazioni espressive che seguono una traiettoria lineare: glissandi, arpeggi e trilli.

Glissando

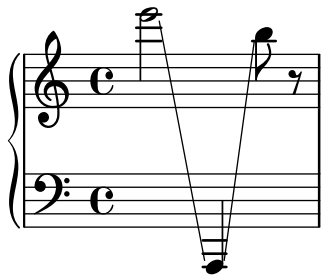
Un *glissando* si crea attaccando `\glissando` a una nota:

```
g2\glissando g'
c2\glissando c,
\afterGrace f,1\glissando f'16
```



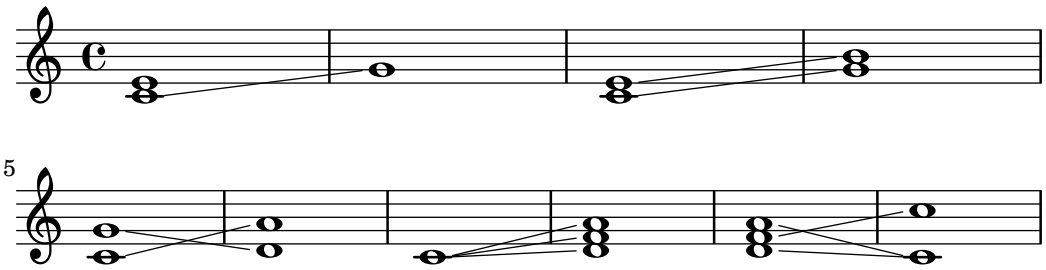
Un glissando può collegare note appartenenti a righi diversi:

```
\new PianoStaff <<
  \new Staff = "right" {
    e'''2\glissando
    \change Staff = "left"
    a,,4\glissando
    \change Staff = "right"
    b''8 r |
  }
  \new Staff = "left" {
    \clef bass
    s1
  }
>>
```



Un glissando può collegare le note negli accordi. Se è necessario qualcosa di diverso dal normale abbinamento uno a uno delle note, si possono definire le connessioni tra le note attraverso `\glissandoMap`, dove le note di un accordo sono numerate a partire da zero nell'ordine in cui appaiono nel file di input '`.ly`'.

```
<c, e>1\glissando g' |
<c, e>1\glissando |
<g' b> |
\break
\set glissandoMap = #'((0 . 1) (1 . 0))
<c, g'>1\glissando |
<d a'> |
\set glissandoMap = #'((0 . 0) (0 . 1) (0 . 2))
c1\glissando |
<d f a> |
\set glissandoMap = #'((2 . 0) (1 . 0) (0 . 1))
<f d a'>1\glissando |
<c c'> |
```



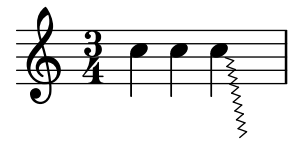
Si possono adottare diversi stili di glissando. Maggiori dettagli in Sezione 5.4.7 [Line styles], pagina 598.

Frammenti di codice selezionati

Glissando contemporaneo

Un glissando contemporaneo senza una nota finale può essere creato usando una nota nascosta e un tempo di cadenza.

```
\relative c'' {
  \time 3/4
  \override Glissando.style = #'zigzag
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
  \bar "|"
}
```



Aggiungere i segni di tempo per i glissandi lunghi

I battiti saltati nei glissandi molto lunghi vengono talvolta segnalati con delle indicazioni di tempo, che consistono solitamente in dei gambi privi di teste di nota. Questi gambi possono essere usati anche per contenere segni di espressione intermedi.

Se i gambi non si allineano bene al glissando, può essere necessario riposizionarli leggermente.

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}
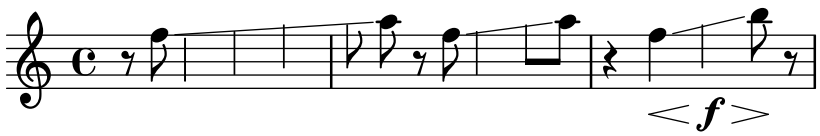
glissandoSkipOff = {
  \revert NoteColumn.glissando-skip
  \undo \hide NoteHead
  \revert NoteHead.no-ledgers
}

\relative c'' {
  r8 f8\glissando
```

```
  \glissandoSkipOn
  f4 g a a8\noBeam
  \glissandoSkipOff
  a8

  r8 f8\glissando
  \glissandoSkipOn
  g4 a8
  \glissandoSkipOff
  a8 |

  r4 f\glissando \<
  \glissandoSkipOn
  a4\f \>
  \glissandoSkipOff
  b8\! r |
}
```

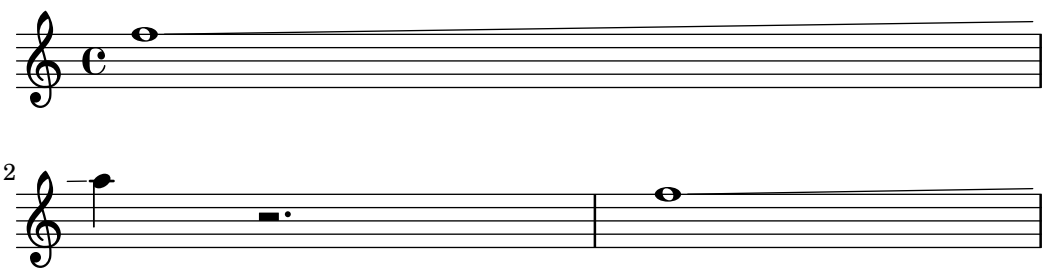


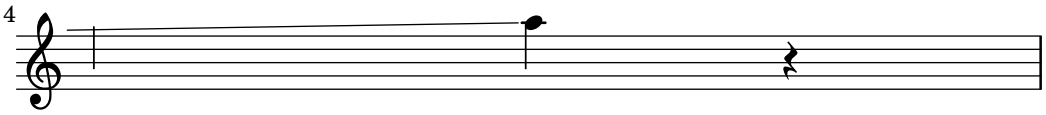
Lasciare che i glissandi vadano a capo

Per permettere a un glissando di andare a capo se capita su un'interruzione di riga, si impostano le proprietà `breakable` e `after-line-breaking` su `#t`:

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

\relative c'' {
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  f1\glissando |
  \break
  a4 r2. |
  f1\glissando
  \once \glissandoSkipOn
  \break
  a2 a4 r4 |
}
```





Estendere i glissandi sulle volte delle ripetizioni

Un glissando che si estende in vari blocchi `\alternative` può essere simulato aggiungendo all'inizio di ogni blocco `\alternative` una nota di abbellimento nascosta da cui inizia un glissando. La nota di abbellimento deve avere la stessa altezza della nota da cui parte il glissando iniziale. In questo frammento si usa una funzione musicale che prende come argomento l'altezza della nota di abbellimento.

Attenzione: nella musica polifonica la nota di abbellimento deve avere una nota di abbellimento corrispondente in tutte le altre voci.

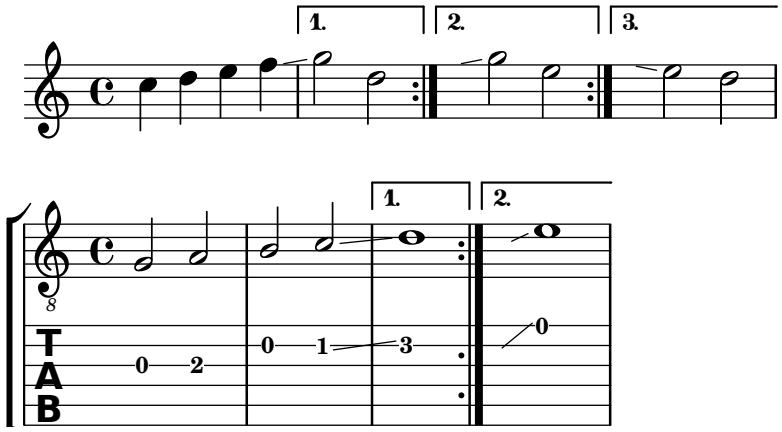
```
repeatGliss = #(define-music-function (parser location grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music =  \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c e1 }
  }
}

\score {
  \new StaffGroup <<
    \new Staff <<
      \context Voice { \clef "G_8" \music }
    >>
    \new TabStaff  <<
      \context TabVoice { \clef "moderntab" \music }
    >>
```

```
  >>
}
```



Vedi anche

Glossario Musicale: Sezione “glissando” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.4.7 [Line styles], pagina 598.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Glissando” in *Guida al Funzionamento Interno*.

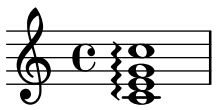
Problemi noti e avvertimenti

Non è supportato il testo lungo la linea del glissando (ad esempio *gliss.*).

Arpeggio

Un *arpeggio* su un accordo (detto anche accordo spezzato) si ottiene aggiungendo `\arpeggio` all’accordo:

```
<c e g c>1\arpeggio
```



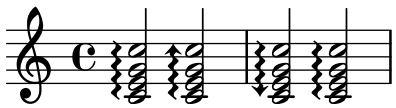
Si possono scrivere vari tipi di arpeggio. `\arpeggioNormal` ripristina l’arpeggio normale:

```
<c e g c>2\arpeggio

\arpeggioArrowUp
<c e g c>2\arpeggio

\arpeggioArrowDown
<c e g c>2\arpeggio

\arpeggioNormal
<c e g c>2\arpeggio
```



Si possono creare simboli di arpeggio speciali *in forma di parentesi*:

```
<c e g c>2

\arpeggioBracket
<c e g c>2\arpeggio

\arpeggioParenthesis
<c e g c>2\arpeggio

\arpeggioParenthesisDashed
<c e g c>2\arpeggio

\arpeggioNormal
<c e g c>2\arpeggio
```



Le proprietà del tratteggio della parentesi dell'arpeggio sono regolate dalla proprietà `'dash-details`, descritta in ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩.

Gli arpeggi possono essere scritti in modo esplicito con le legature di valore. Per maggiori dettagli si veda ⟨undefined⟩ [Ties], pagina ⟨undefined⟩.

Comandi predefiniti

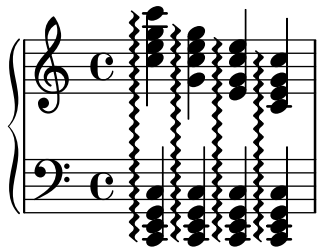
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis` `\arpeggioParenthesisDashed`.

Frammenti di codice selezionati

Creare degli arpeggi che attraversano il rigo del pianoforte

In un rigo per pianoforte (`PianoStaff`), è possibile far sì che un arpeggio attraversi i righi impostando la proprietà `PianoStaff.connectArpeggios`.

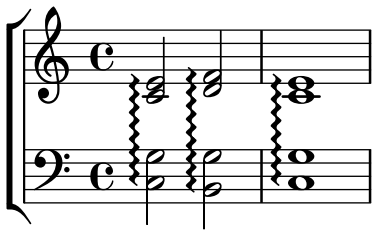
```
\new PianoStaff \relative c'' <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff {
    <c e g c>4\arpeggio
    <g c e g>4\arpeggio
    <e g c e>4\arpeggio
    <c e g c>4\arpeggio
  }
  \new Staff {
    \clef bass
    \repeat unfold 4 {
      <c,, e g c>4\arpeggio
    }
  }
>>
```



Creare degli arpeggi che attraversano i righi in altri contesti

Si possono creare arpeggi che attraversano i righi in contesti diversi da `GrandStaff`, `PianoStaff` e `StaffGroup` se l'incisore `Span_arpeggio_engraver` è incluso nel contesto `Score`.

```
\score {
  \new ChoirStaff {
    \set Score.connectArpeggios = ##t
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
  }
  \layout {
    \context {
      \Score
      \consists "Span_arpeggio_engraver"
    }
  }
}
```
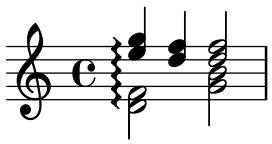


Creare degli arpeggi che attraversano note appartenenti a voci diverse

Si può disegnare un arpeggio che attraversa delle note in voci diverse dello stesso rigo se si aggiunge l'incisore `Span_arpeggio_engraver` nel contesto `Staff`:

```
\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' {
  \set Staff.connectArpeggios = ##t
  <<
    { <e' g>4\arpeggio <d f> <d f>2 }
```

```
    \\
    { <d, f>2\arpeggio <g b>2 }
  >>
}
```



Vedi anche

Glossario Musicale: Sezione “arpeggio” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩, ⟨undefined⟩ [Ties], pagina ⟨undefined⟩.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Arpeggio” in *Guida al Funzionamento Interno*, Sezione “Slur” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Non è possibile mostrare simultaneamente arpeggi connessi e non connessi in un `PianoStaff`.

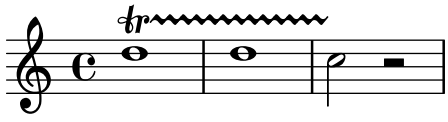
L’arpeggio in forma di parentesi non può essere impostato con facilità negli arpeggi che attraversano i righi; occorre ricorrere a metodi più complessi descritti in [Cross-staff stems], pagina 315.

Trilli

I trilli senza linea di estensione si ottengono col comando `\trill`; si veda ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.

I trilli con linea di estensione si ottengono con `\startTrillSpan` e `\stopTrillSpan`:

```
d1\startTrillSpan
d1
c2\stopTrillSpan
r2
```



Un estensore del trillo che va a capo ricomincerà esattamente sopra la prima nota della nuova riga.

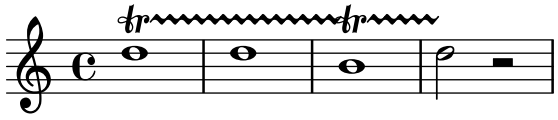
```
d1\startTrillSpan
\break
d1
c2\stopTrillSpan
r2
```
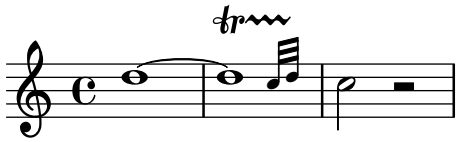




È possibile tracciare trilli consecutivi senza dover esplicitare i comandi `\stopTrillSpan`, perché il trillo successivo diventerà automaticamente il limite destro di quello precedente.

```
d1\startTrillSpan
d1
b1\startTrillSpan
d2\stopTrillSpan
r2
```
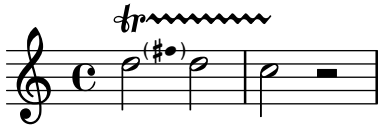


I trilli possono essere anche combinati con le note di abbellimento. La sintassi di questo costrutto e il metodo per posizionare in modo preciso gli abbellimenti sono descritti in ⟨undefined⟩ [Grace notes], pagina ⟨undefined⟩.

```
d1~\afterGrace
d1\startTrillSpan { c32[ d]\stopTrillSpan }
c2 r2
```



I trilli che richiedono una nota ausiliaria dall'altezza esplicita si ottengono col comando `\pitchedTrill`. Il primo argomento è la nota principale e il secondo è la nota *trillata*, che appare come una testa di nota senza gambo e racchiusa tra parentesi.
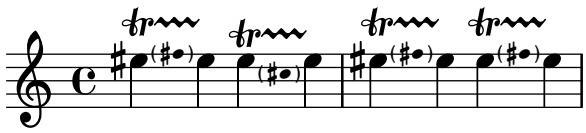
```
\pitchedTrill
d2\startTrillSpan fis
d2
c2\stopTrillSpan
r2
```



Alterazioni successive della stessa nota nella stessa misura devono essere aggiunte manualmente. Apparirà solo l'alterazione del primo trillo con notina in una misura.

```
\pitchedTrill
eis4\startTrillSpan fis
eis4\stopTrillSpan
\pitchedTrill
eis4\startTrillSpan cis
eis4\stopTrillSpan
\pitchedTrill
eis4\startTrillSpan fis
eis4\stopTrillSpan
\pitchedTrill
```

```
eis4\startTrillSpan fis!
eis4\stopTrillSpan
```



Comandi predefiniti

`\startTrillSpan`, `\stopTrillSpan`.

Vedi anche
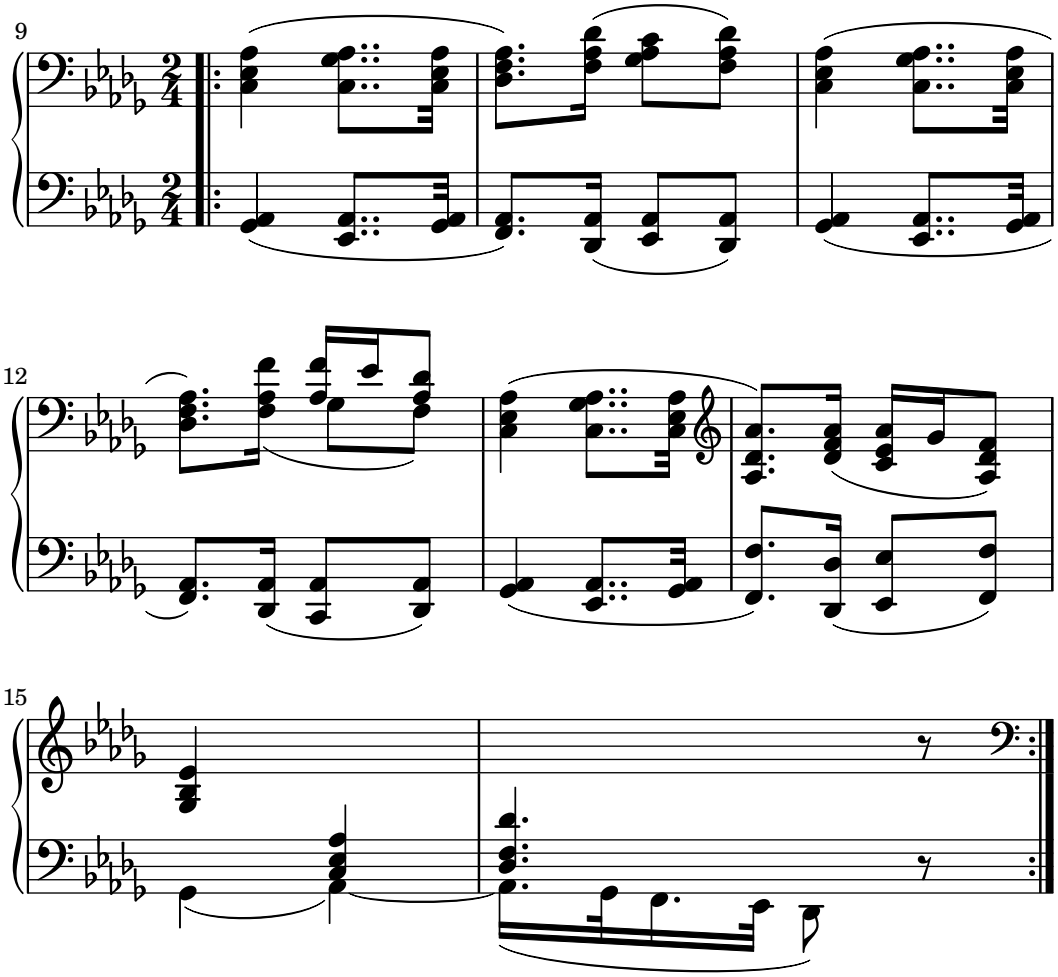
Glossario Musicale: Sezione “trillo” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩, ⟨undefined⟩ [Grace notes], pagina ⟨undefined⟩.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TrillSpanner” in *Guida al Funzionamento Interno*.

1.4 Ripetizioni



La ripetizione è un concetto chiave in musica e può essere resa con varie forme di notazione. LilyPond supporta i seguenti tipi di ripetizioni:

`volta` La musica ripetuta non viene scritta per intero ma racchiusa tra barre di ripetizione. Se la ripetizione si trova all'inizio di un brano, la stanghetta di ritornello è posta soltanto alla fine della ripetizione. I finali alternativi (volte) appaiono da sinistra a destra e sono evidenziati da delle parentesi. Questa è la notazione standard per le ripetizioni con finali alternativi.

`unfold` La musica ripetuta viene scritta per intero, tante volte quante sono specificate dal *`numero-ripetizioni`*. È utile quando si scrive musica ripetitiva.

`percent` Si tratta di ripetizioni del singolo tempo (battito) o della battuta. Hanno l'aspetto di una barra obliqua o di segni di percentuale.

`tremolo` Si usa per scrivere travature a tremolo.

1.4.1 Ripetizioni lunghe

Questa sezione spiega come inserire ripetizioni lunghe (solitamente di più battute). Tali ripetizioni possono essere in due forme: racchiuse tra segni di ritornello oppure ricopiate interamente (adatte a scrivere musica ripetitiva). Si possono anche controllare manualmente i segni di ripetizione.

Ripetizioni normali
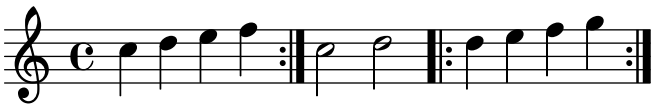
La sintassi per una normale ripetizione è

```
\repeat volta numero-di-ripetizioni espressione-musicale
```

dove *`espressione-musicale`* è la musica da ripetere.

Un'unica ripetizione senza finale alternativo:

```
\repeat volta 2 { c4 d e f }
c2 d
\repeat volta 2 { d4 e f g }
```



I finali alternativi si ottengono con `\alternative`. Ogni gruppo di alternative deve essere a sua volta racchiuso tra parentesi.
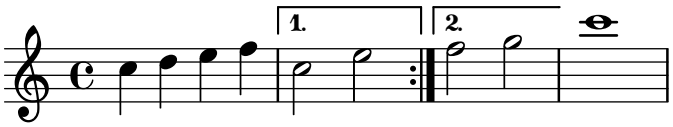
```
\repeat volta numero-di-ripetizioni espressione-musicale
\alternative {
  { espressione-musicale }
}
```

dove *`espressione-musicale`* è la musica.

Se il numero di ripetizioni è superiore a quello dei finali alternativi, alle prime ripetizioni viene assegnata la prima alternativa.
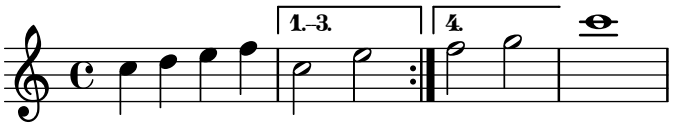
Una singola ripetizione con un finale alternativo:

```
\repeat volta 2 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
}
c1
```



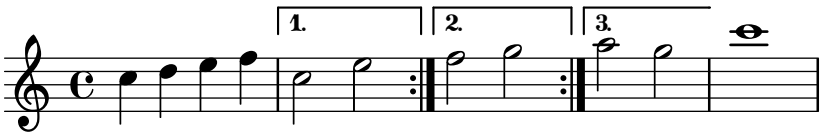
Molteplici ripetizioni con un finale alternativo:

```
\repeat volta 4 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
}
c1
```



Molteplici ripetizioni con più di un finale alternativo:

```
\repeat volta 3 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
  { a2 g | }
}
c1
```
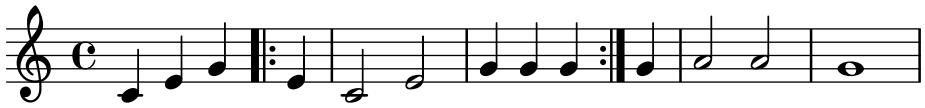


Nota: Se ci sono due o più finali alternativi, non ci deve essere niente tra la parentesi di chiusura di uno e quella di apertura di quello successivo all'interno di un blocco `\alternative`, altrimenti non si otterrà il numero atteso di finali.

Nota: Se si usa `\relative` dentro a un blocco `\repeat` senza istanziare esplicitamente il contesto `Voice`, appare un rigo in più (non desiderato). Vedi Sezione "Appare un rigo in più" in *Uso del Programma*.

Se una ripetizione inizia in mezzo a una misura e non ha finali alternativi, solitamente anche la chiusura della ripetizione cadrà nel mezzo di una misura, così che tra le due estremità ci sia una misura completa. In queste situazioni i segni di ripetizione non sono delle vere e proprie stanghette. Non usare il comando `\partial` o i controlli di battuta nel punto in cui si trovano questi segni:
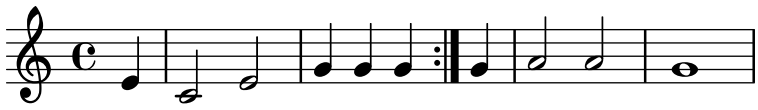
```
% nessun \partial qui
c4 e g  % nessun controllo di battuta qui
% nessun \partial qui
\repeat volta 4 {
  e4 |
  c2 e |
  % nessun \partial qui
  g4 g g  % nessun controllo di battuta qui
```

```
}
% nessun \partial qui
g4 |
a2 a |
g1 |
```



Analogalmente, se una ripetizione inizia con la misura parziale a inizio della partitura e non ha finali alternativi, si applicano le stesse condizioni dell'esempio precedente, a parte il fatto che in questo caso è richiesto il comando `\partial` all'inizio della partitura:
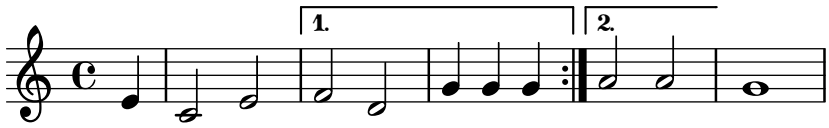
```
\partial 4  % necessario
\repeat volta 4 {
  e4 |
  c2 e |
  % nessun \partial qui
  g4 g g  % nessun controllo di battuta qui
}
% nessun \partial qui
g4 |
a2 a |
g1 |
```



Quando vengono aggiunti dei finali alternativi a una ripetizione che inizia con una misura incompleta, diventa necessario impostare manualmente la proprietà di contesto `Timing.measureLength` nei seguenti punti specifici:

- all'inizio di tutte le misure incomplete nel blocco `\alternative`, che di solito si trovano alla fine di ogni alternativa, con l'eccezione dell'ultima (nella maggior parte dei casi).
- all'inizio di ogni alternativa, eccetto la prima.

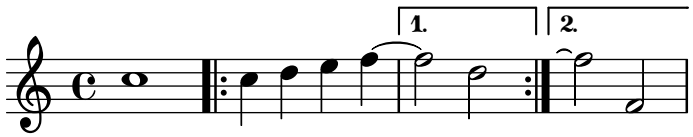
```
\partial 4
\repeat volta 2 { e4 | c2 e | }
\alternative {
  {
    f2 d |
    \set Timing.measureLength = #(ly:make-moment 3/4)
    g4 g g  % un controllo di battuta opzionale è permesso qui
  }
  {
    \set Timing.measureLength = #(ly:make-moment 4/4)
    a2 a |
  }
}
g1 |
```



La proprietà `measureLength` è descritta in ⟨undefined⟩ [Time administration], pagina ⟨undefined⟩.

Si possono aggiungere delle legature di valore a un secondo finale:
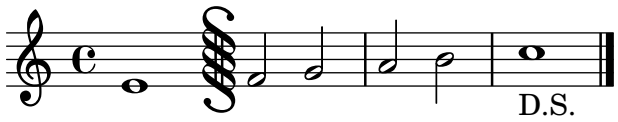
```
c1
\repeat volta 2 { c4 d e f~ }
\alternative {
  { f2 d }
  { f2\repeatTie f, }
}
```



Il comando `\inStaffSegno` può essere usato per generare una stanghetta composita che incorpora il simbolo di segno nella stanghetta di ripetizione appropriata se usato col comando `\repeat volta`. Il tipo corretto di stanghetta di ripetizione, ovvero inizio della ripetizione, fine della ripetizione e doppia ripetizione, viene selezionato automaticamente. Il corrispondente segno "D.S." deve essere agiunto manualmente.
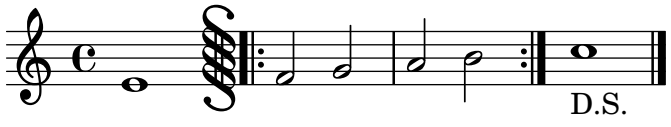
Lontano da una ripetizione:

```
e1
\inStaffSegno
f2 g a b
c1_"D.S." \bar "|."
```



All'inizio di una ripetizione:

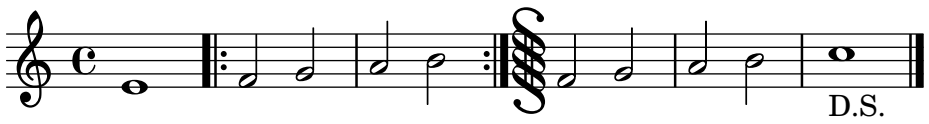
```
e1
\repeat volta 2 {
  \inStaffSegno  % start repeat
  f2 g a b
}
c1_"D.S." \bar "|."
```



Alla fine di una ripetizione:

```
e1
\repeat volta 2 {
  f2 g a b
  \inStaffSegno  % end repeat
}
f2 g a b
```

```
c1_"D.S." \bar "|."
```



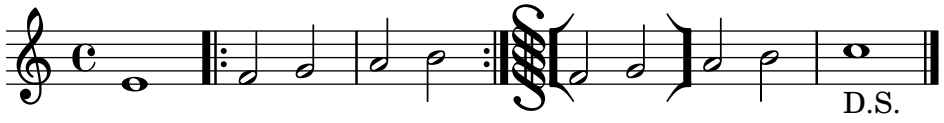
Tra due ripetizioni:

```
e1
\repeat volta 2 {
  f2 g a b
}
\inStaffSegno  % double repeat
\repeat volta 2 {
  f2 g a b
}
c1_"D.S." \bar "|."
```
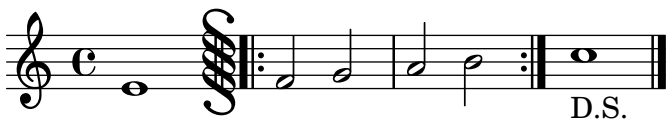


Si possono impostare simboli alternativi delle stanghette modificando nel contesto Score le proprietà `segnoType`, `startRepeatSegnoType`, `endRepeatSegnoType` o `doubleRepeatSegnoType` per il tipo di stanghetta richiesto. I tipi di stanghetta alternativi devono essere selezionati dai tipi predefiniti o dai tipi precedentemente definiti col comando `\defineBarLine` (vedi ⟨undefined⟩ [Bar lines], pagina ⟨undefined⟩).

```
\defineBarLine ":|.S[" #'(":|." "S[" "")
\defineBarLine "]" #'("]" "" "")
e1
\repeat volta 2 {
  f2 g a b
  \once \set Score.endRepeatSegnoType = ":|.S["
  \inStaffSegno
}
f2 g \bar "]" a b
c1_"D.S." \bar "|."
```




```
e1
\repeat volta 2 {
  \inStaffSegno
  f2 g a b
}
c1_"D.S." \bar "|."
```
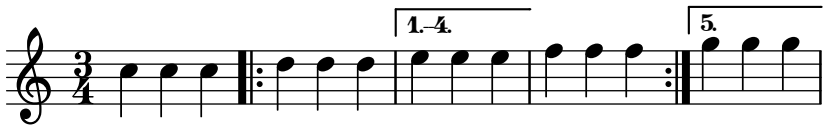


Frammenti di codice selezionati

Accorciare le parentesi delle volte

Per impostazione predefinita, le parentesi delle volte si estendono per tutta l'alternativa, ma si possono accorciare impostando `voltaSpannerDuration`. Nell'esempio seguente, la parentesi dura una misura, che ha una durata di 3/4.

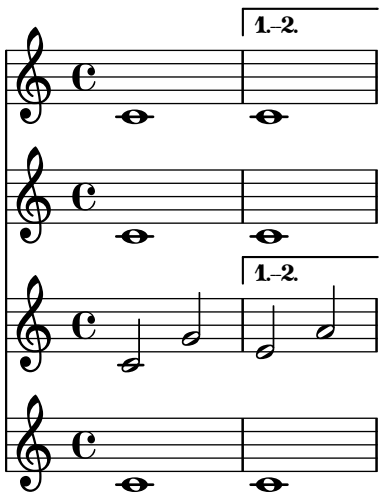
```
\relative c'' {
  \time 3/4
  c4 c c
  \set Score.voltaSpannerDuration = #(ly:make-moment 3/4)
  \repeat volta 5 { d4 d d }
  \alternative {
    {
      e4 e e
      f4 f f
    }
    { g4 g g }
  }
}
```



Aggiungere le parentesi delle volte a altri righi

L'incisore `Volta_engraver` risiede nel contesto `Score`, quindi le parentesi delle ripetizioni appaiono di norma soltanto sul rigo superiore. Questo comportamento può essere modificato aggiungendo l'incisore `Volta_engraver` al contesto `Staff` in cui si desidera far apparire le parentesi; si veda anche il frammento "Volta multirigo".

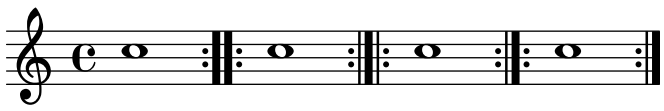
```
<<
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
>>
```



Impostare la doppia ripetizione predefinita per le volte

Esistono tre diversi stili di doppie ripetizioni per le volte, che si possono impostare con `doubleRepeatType`.
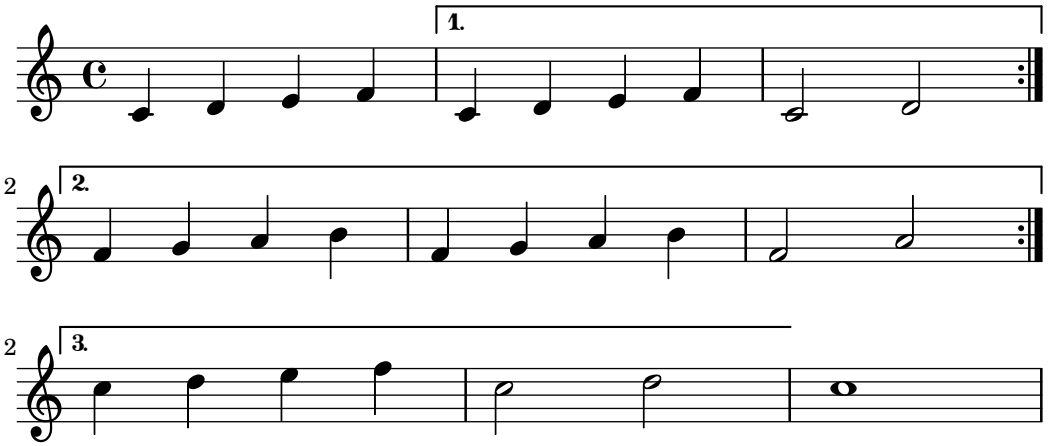
```
\relative c'' {
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":..:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.:"
  \repeat volta 1 { c1 }
}
```
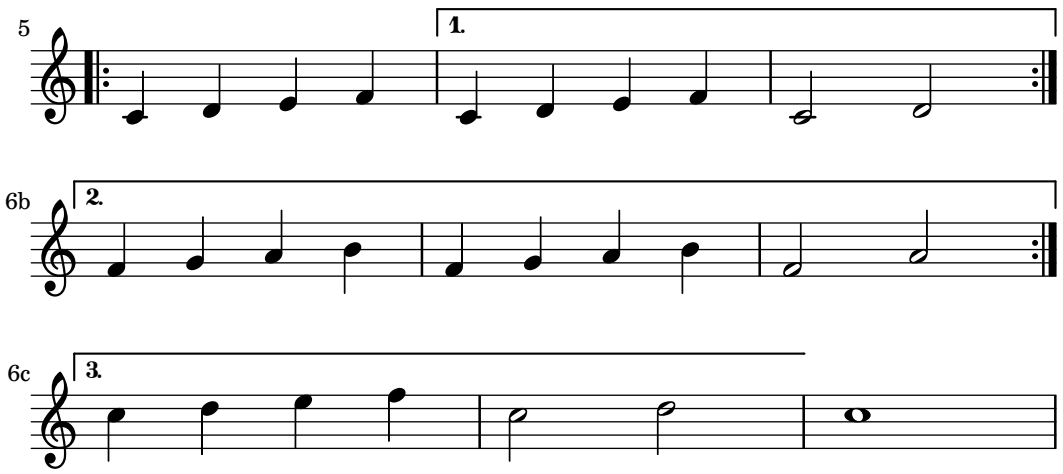


Numeri di battuta alternativi

Si possono impostare due metodi alternativi di numerazione della battuta, utili specialmente per le ripetizioni.

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```





Vedi anche

Glossario Musicale: Sezione “ripetizione” in *Glossario Musicale*, Sezione “volta” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Bar lines], pagina ⟨undefined⟩, Sezione 5.1.4 [Modifying context plug-ins], pagina 561, [Modifying ties and slurs], pagina 605, ⟨undefined⟩ [Time administration], pagina ⟨undefined⟩.

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VoltaBracket” in *Guida al Funzionamento Interno*, Sezione “RepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “VoltaRepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “UnfoldedRepeatedMusic” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Le legature di portamento che si estendono da un blocco `\repeat` verso un blocco `\alternative` funzioneranno solo nel primo finale alternativo. L’aspetto grafico di una legatura di portamento che continua negli altri finali alternativi può essere simulato con `\repeatTie` se la legatura si estende solo su una nota del blocco dell’alternativa, sebbene questo metodo non funzioni in `TabStaff`. Altri metodi che si possono adattare per indicare legature di portamento che continuano su varie note dei blocchi di alternativa, e che funzionano anche nei contesti `TabStaff`, sono presentati in [Modifying ties and slurs], pagina 605.

Inoltre le legature di portamento non possono ricollegarsi dalla fine di un’alternativa all’inizio della ripetizione.

I glissandi che si estendono da un blocco `\repeat` in un blocco `\alternative` funzioneranno soltanto per il primo finale alternativo. L’aspetto grafico di un glissando che continua negli altri finali alternativi può essere indicato creando un glissando che inizia su una nota di abbellimento nascosta. Vedere ad esempio il frammento “Estendere i glissandi attraverso le ripetizioni” nei Frammenti Selezionati in [Glissando], pagina 133.

Se una ripetizione che inizia con una misura incompleta ha un blocco `\alternative` che contiene modifiche alla proprietà `measureLength`, l’uso di `\unfoldRepeats` causerà l’erroneo posizionamento delle stanghette e degli avvisi di controllo di battuta.

Una ripetizione annidata come la seguente

```
\repeat ...
\repeat ...
\alternative
```

è ambigua, perché non è chiaro a quale `\repeat` appartenga il blocco `\alternative`. Questa ambiguità si risolve facendo in modo che `\alternative` appartenga sempre al blocco `\repeat` interno. Per chiarezza, si consiglia di usare le parentesi in queste situazioni.
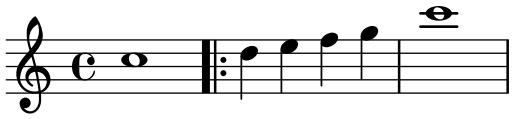
Indicazioni di ripetizione manuali

> **Nota:** Questi metodi vengono usati solo per mostrare tipi di ripetizioni inusuali, e potrebbero causare un comportamento inaspettato. Nella maggior parte dei casi, le ripetizioni devono essere create col comando standard `\repeat` oppure stampando le stanghette opportune. Maggiori informazioni in ⟨undefined⟩ [Bar lines], pagina ⟨undefined⟩.

La proprietà `repeatCommands` permette di controllare la formattazione delle ripetizioni. Il suo valore è una lista Scheme dei comandi di ripetizione.

`start-repeat`
: Stampa una stanghetta `.|:`.
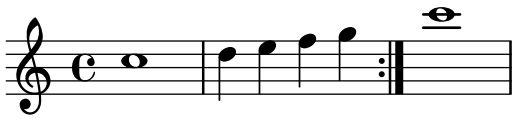
```
c1
\set Score.repeatCommands = #'(start-repeat)
d4 e f g
c1
```



Come vuole la pratica comune di incisione, i segni di ripetizione non vengono stampati all'inizio di un brano.
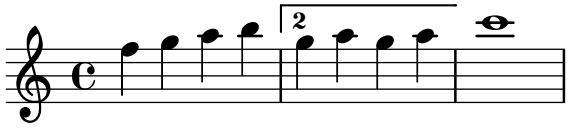
`end-repeat`
: Stampa una stanghetta `:|.`:

```
c1
d4 e f g
\set Score.repeatCommands = #'(end-repeat)
c1
```



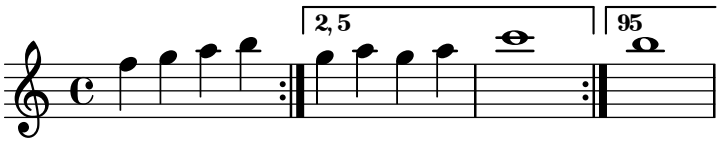
`(volta` *numero*`) ... (volta #f)`
: Crea una nuova volta col numero specificato. La parentesi della volta deve essere terminata esplicitamente, altrimenti non sarà stampata.

```
f4 g a b
\set Score.repeatCommands = #'((volta "2"))
g4 a g a
\set Score.repeatCommands = #'((volta #f))
c1
```
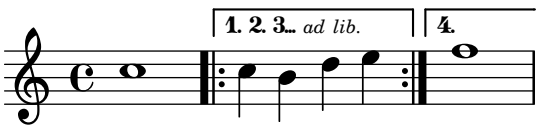


Comandi di ripetizione multipli possono trovarsi nello stesso punto:

```
f4 g a b
\set Score.repeatCommands = #'((volta "2, 5") end-repeat)
g4 a g a
c1
\set Score.repeatCommands = #'((volta #f) (volta "95") end-repeat)
b1
\set Score.repeatCommands = #'((volta #f))
```



Si può includere del testo nella parentesi della volta. Il testo può consistere di un numero, di più numeri o di un'indicazione testuale, si veda ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩. Il modo più semplice per usare del testo è definirlo prima e poi includerlo nella lista Scheme,

```
voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
\relative c'' {
  c1
  \set Score.repeatCommands =
    #(list(list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}
```



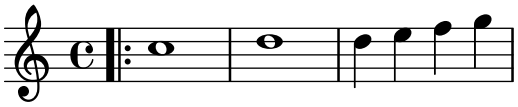
Frammenti di codice selezionati

Stampare un segno di ripetizione all'inizio di un brano

Una stanghetta `.|:` può apparire all'inizio di un brano, se si sovrascrive la proprietà pertinente:

```
\relative c'' {
  \once \override Score.BreakAlignment.break-align-orders =
    #(make-vector 3 '(instrument-name
                      left-edge
                      ambitus
                      breathing-sign
                      clef
                      key-signature
                      time-signature
                      staff-bar
                      custos))
  \once \override Staff.TimeSignature.space-alist =
    #'((first-note . (fixed-space . 2.0))
       (right-edge . (extra-space . 0.5))
```

```
        ;; free up some space between time signature
        ;; and repeat bar line
        (staff-bar . (extra-space . 1)))
  \bar ".|:"
  c1
  d1
  d4 e f g
}
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Bar lines], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VoltaBracket” in *Guida al Funzionamento Interno*, Sezione “RepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “VoltaRepeatedMusic” in *Guida al Funzionamento Interno*.
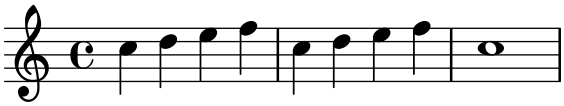
Ripetizioni ricopiate

Col comando `unfold`, le ripetizioni possono servire a semplificare la scrittura di musica ripetitiva. La sintassi è

```
\repeat unfold numero-di-ripetizioni espressione-musicale
```

dove *`espressione-musicale`* è la musica e *`numero-di-ripetizioni`* è il numero di volte per cui è ripetuta *`espressione-musicale`*.

```
\repeat unfold 2 { c4 d e f }
c1
```



In alcuni casi, specialmente in un contesto `\relative`, la funzione `\repeat unfold` non equivale a riscrivere l’espressione musicale più volte. Ad esempio
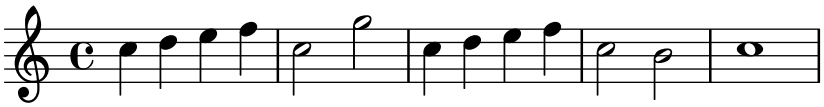
```
\repeat unfold 2 { a'4 b c }
```

non equivale a

```
a'4 b c | a'4 b c
```

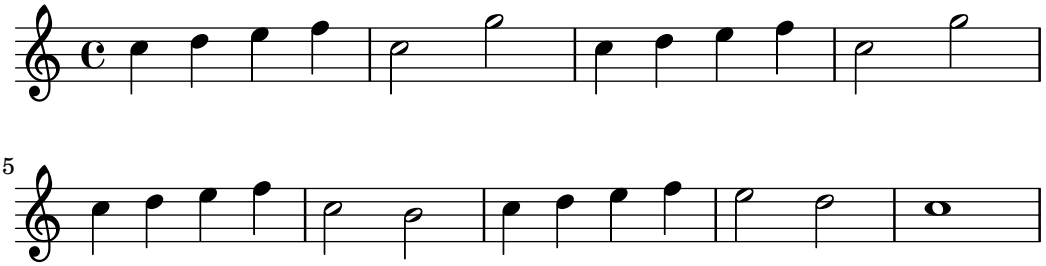
Le ripetizioni dispiegate (unfold) possono avere dei finali alternativi.

```
\repeat unfold 2 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
}
c1
```
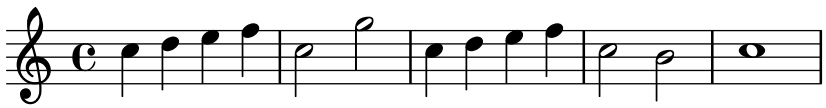


Se il numero di ripetizioni è maggiore del numero di finali alternativi, la prima alternativa viene applicata più volte, finché le alternative rimaste non esauriscono il numero totale delle ripetizioni.

```
\repeat unfold 4 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
  { e2 d }
 }
c1
```



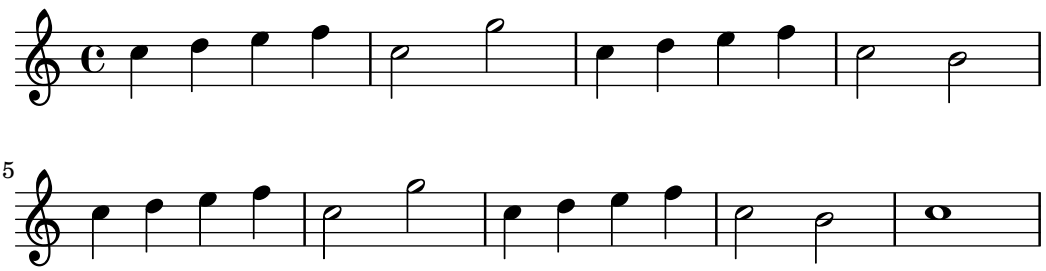
Se il numero di finali alternativi è maggiore del numero di ripetizioni, solo le prime alternative vengono applicate. Le alternative rimanenti saranno ignorate e non verranno stampate.

```
\repeat unfold 2 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
  { e2 d }
}
c1
```



È anche possibile annidare molteplici funzioni `unfold` (con o senza finali alternativi).

```
\repeat unfold 2 {
  \repeat unfold 2 { c4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
  }
}
c1
```



Gli accordi si ripetono col simbolo di ripetizione dell'accordo `q`. Vedi ⟨undefined⟩ [Chord repetition], pagina ⟨undefined⟩.

> **Nota:** Se si usa `\relative` dentro a un blocco `\repeat` senza istanziare esplicitamente il contesto `Voice`, appare un rigo in più (non desiderato). Vedi Sezione "Appare un rigo in più" in *Uso del Programma*.

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Chord repetition], pagina ⟨undefined⟩.

Frammenti: Sezione "Repeats" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "RepeatedMusic" in *Guida al Funzionamento Interno*, Sezione "UnfoldedRepeatedMusic" in *Guida al Funzionamento Interno*.

1.4.2 Ripetizioni brevi

Questa sezione tratta il modo in cui inserire brevi ripetizioni. Le ripetizioni brevi possono avere due forme: segni di tratto obliquo o percentuale per rappresentare le ripetizioni di una singola nota, di una singola misura o di due misure; tremolo negli altri casi.

Ripetizioni con percentuale

Brevi sezioni ripetute vengono stampate la prima volta volta e le ripetizioni vengono sostituite da un apposito segno.

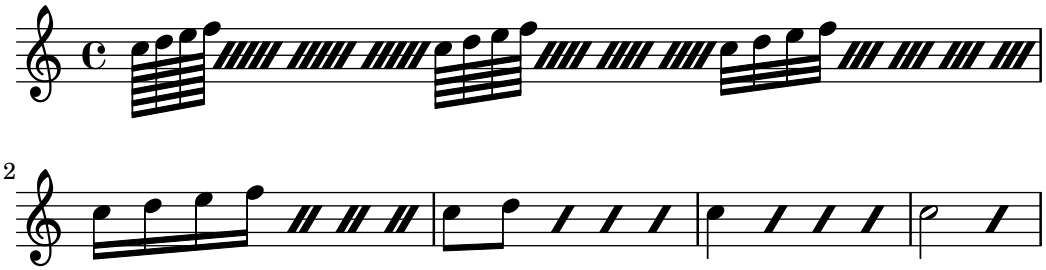
La sintassi è

```
\repeat percent numero espressione-musicale
```

dove *`espressione-musicale`* è l'espressione musicale da ripetere.
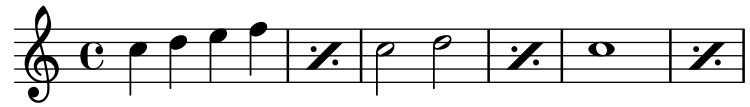
Fraseggi più brevi di una misura vengono sostituiti dal tratto obliquo.

```
\repeat percent 4 { c128 d e f }
\repeat percent 4 { c64 d e f }
\repeat percent 5 { c32 d e f }
\repeat percent 4 { c16 d e f }
\repeat percent 4 { c8 d }
\repeat percent 4 { c4 }
\repeat percent 2 { c2 }
```
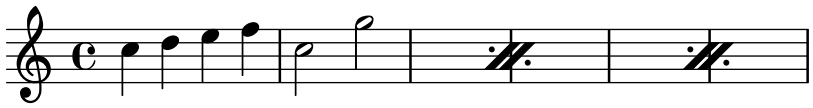


Fraseggi di una o due misure vengono sostituiti da simboli simili alla percentuale.

```
\repeat percent 2 { c4 d e f }
\repeat percent 2 { c2 d }
\repeat percent 2 { c1 }
```

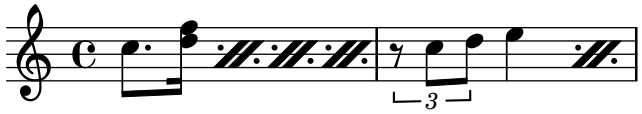


```
\repeat percent 3 { c4 d e f | c2 g' }
```



Fraseggi più brevi di una misura ma con durate miste adottano un simbolo di doppia percentuale.

```
\repeat percent 4 { c8. <d f>16 }
\repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
```

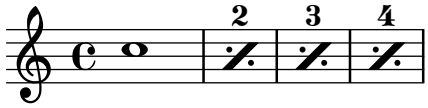


Frammenti di codice selezionati

Contatore della ripetizione con segno percentuale

Le ripetizioni di misura che hanno più di due ripetizioni possono avere un contatore se si cambia la proprietà opportuna, come mostra questo esempio:

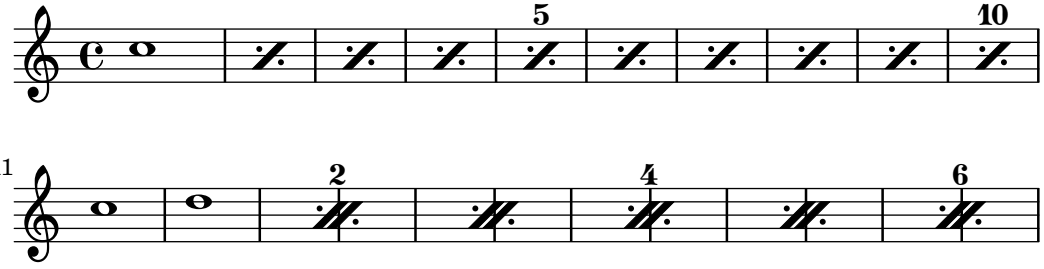
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



Visibilità del conto della ripetizione con segno percentuale

I contatori della ripetizione con segno percentuale possono essere mostrati a intervalli regolari impostando la proprietà di contesto `repeatCountVisibility`.

```
\relative c'' {
  \set countPercentRepeats = ##t
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)
  \repeat percent 10 { c1 } \break
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
  \repeat percent 6 { c1 d1 }
}
```
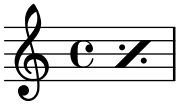


Ripetizioni con segni di percentuale isolati

Si possono stampare anche segni di percentuale isolati.

```
makePercent =
#(define-music-function (parser location note) (ly:music?)
   "Make a percent repeat the same length as NOTE."
   (make-music 'PercentEvent
               'length (ly:music-length note)))

\relative c'' {
  \makePercent s1
}
```



Vedi anche

Glossario Musicale: Sezione “percent repeat” in *Glossario Musicale*, Sezione “simile” in *Glossario Musicale*.

Frammenti: Sezione “Repeats” in *Frammenti di codice*.
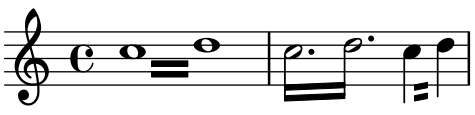
Guida al funzionamento interno: Sezione “RepeatSlash” in *Guida al Funzionamento Interno*, Sezione “RepeatSlashEvent” in *Guida al Funzionamento Interno*, Sezione “DoubleRepeatSlash” in *Guida al Funzionamento Interno*, Sezione “PercentRepeat” in *Guida al Funzionamento Interno*, Sezione “PercentRepeatCounter” in *Guida al Funzionamento Interno*, Sezione “PercentRepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “Percent_repeat_engraver” in *Guida al Funzionamento Interno*, Sezione “DoublePercentEvent” in *Guida al Funzionamento Interno*, Sezione “DoublePercentRepeat” in *Guida al Funzionamento Interno*, Sezione “DoublePercentRepeatCounter” in *Guida al Funzionamento Interno*, Sezione “Double_percent_repeat_engraver” in *Guida al Funzionamento Interno*, Sezione “Slash_repeat_engraver” in *Guida al Funzionamento Interno*.

Ripetizioni con tremolo

I tremoli possono avere due forme: alternanza tra due note, o due accordi, e rapida ripetizione di una singola nota o accordo. I tremoli costituiti da un’alternanza si indicano con delle travature che collegano le note o gli accordi che si alternano, mentre i tremoli che consistono in una rapida ripetizione di una nota singola si indicano aggiungendo delle travature o dei tratti di suddivisione obliqui alla singola nota.

Per inserire i segni del tremolo tra le note, si usa `\repeat` con lo stile tremolo:

```
\repeat tremolo 8 { c16 d }
\repeat tremolo 6 { c16 d }
\repeat tremolo 2 { c16 d }
```
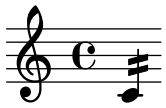


La sintassi di `\repeat tremolo` prevede specificamente che all’interno delle parentesi siano indicate due note, e che il numero di ripetizioni corrisponda a un valore espresso in durate di note normali o puntate. Dunque `\repeat tremolo 7` è valido e produce una nota doppiamente puntata, mentre `\repeat tremolo 9` non è valido.

La durata del tremolo equivale alla durata dell’espressione musicale tra parentesi moltiplicata per il numero di ripetizioni: `\repeat tremolo 8 { c16 d16 }` corrisponde a un tremolo di una semibreve, rappresentata come due semibrevi unite dalle travature del tremolo.
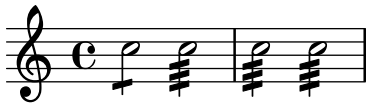
Ci sono due modi di inserire dei segni di tremolo su una singola nota. Anche in questo caso si usa la sintassi `\repeat tremolo`, ma la nota non deve essere racchiusa tra parentesi:

```
\repeat tremolo 4 c'16
```



Si può ottenere lo stesso output aggiungendo `:N` dopo la nota, dove *N* indica la durata della suddivisione (deve essere almeno 8). Se *N* è 8, viene aggiunta una travatura al gambo della nota. Se *N* è omesso, viene usato l'ultimo valore (salvato in `tremoloFlags`):

```
c2:8 c:32
c: c:
```

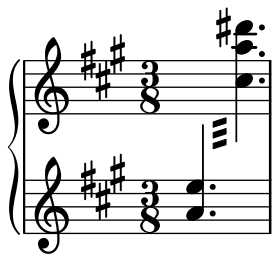


Frammenti di codice selezionati

Tremoli attraverso i righi

Dato che `\repeat tremolo` si aspetta esattamente due argomenti musicali per i tremoli di accordi, la nota o l'accordo che cambiano rigo in un tremolo che attraversa i righi devono essere posti tra parentesi graffe insieme al comando `\change Staff`.
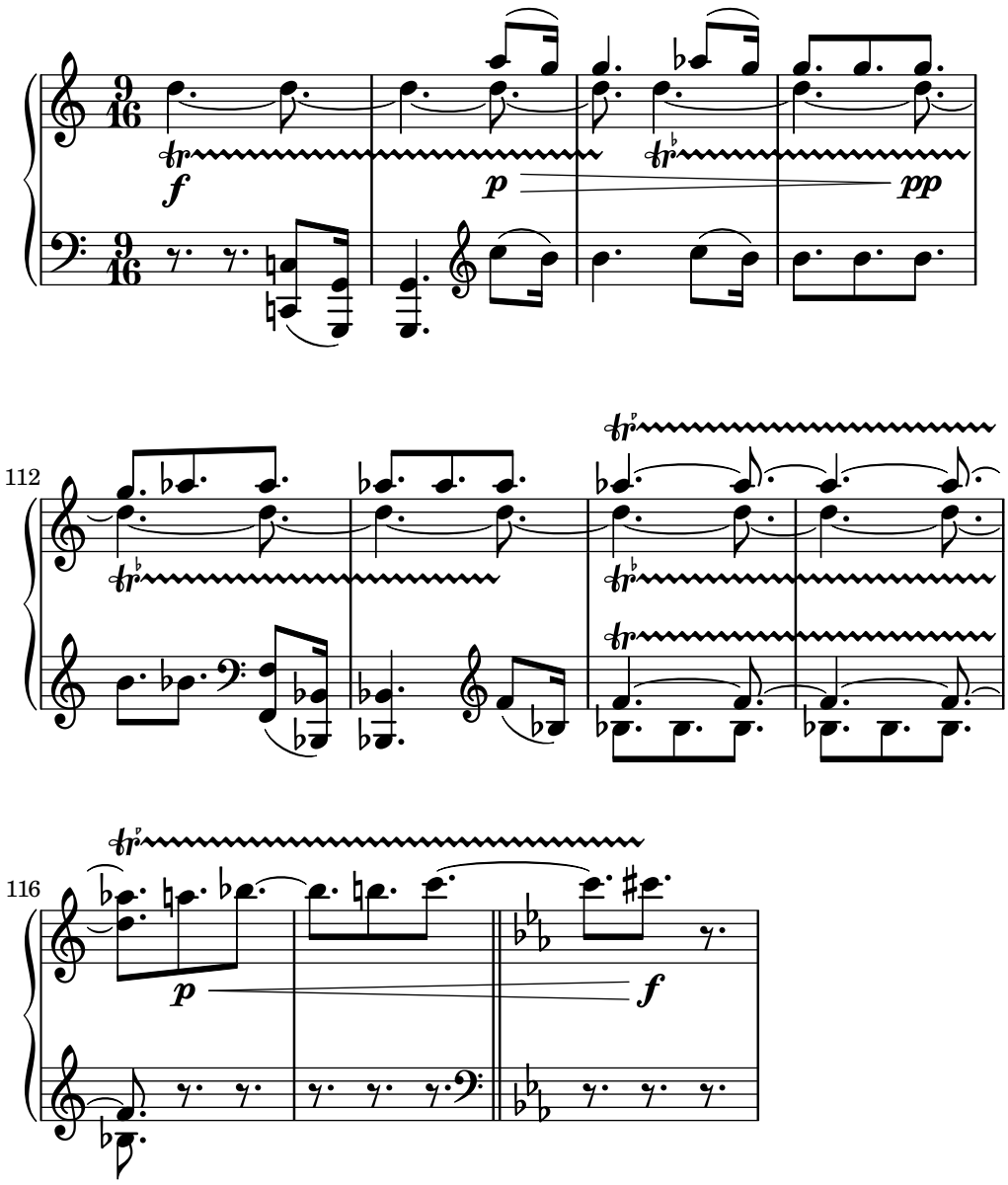
```
\new PianoStaff <<
  \new Staff = "up" \relative c'' {
    \key a \major
    \time 3/8
    s4.
  }
  \new Staff = "down" \relative c'' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
>>
```



Vedi anche

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

1.5 Note simultanee



La polifonia in musica si riferisce alla coesistenza simultanea di più di una voce in un brano musicale. In LilyPond la polifonia si riferisce alla coesistenza di più voci sullo stesso rigo.

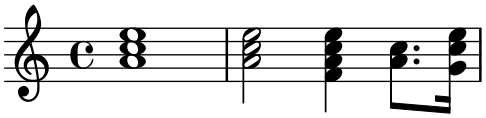
1.5.1 Una voce

In questa sezione vengono spiegate le note simultanee che fanno parte di un’unica voce.

Note in un accordo

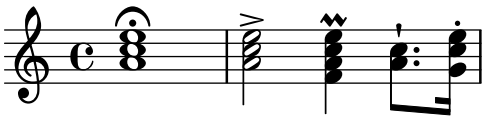
Un accordo si forma racchiudendo una serie di altezze tra `<` e `>` e può essere seguito da una durata, come accade per le semplici note.

```
<a c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
```



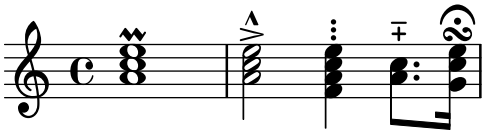
Proprio come per le note, si possono specificare le articolazioni da riferire all'accordo.

```
<a c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^! <g c e>16-.
```



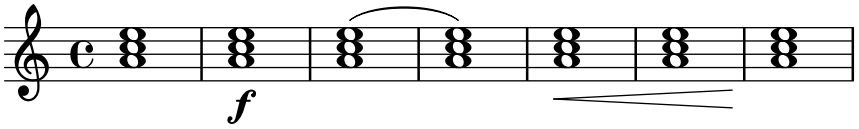
Si possono specificare abbellimenti e articolazioni per ogni nota che fa parte dell'accordo.

```
<a c\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4
<a-+ c-->8. <g\fermata c e\turn>16
```
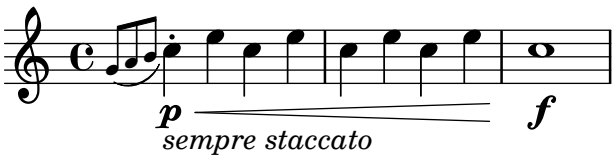


Tuttavia, alcuni elementi della notazione, come le dinamiche, le forcelle e le legature di portamento, devono essere attaccate all'accordo invece che alle sue singole note, altrimenti non appariranno.

```
<a\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>)
<a c e>\< <a c e> <a c e>\!
```



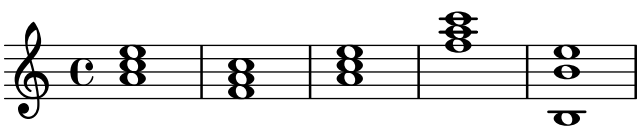
Un accordo si comporta semplicemente come un contenitore di note, articolazioni e altri elementi. Di conseguenza, un accordo privo di note non ha una durata. Qualsiasi articolazione attaccata a un accordo vuoto si troverà nel momento musicale della nota o accordo seguenti e si combinerà con questi (possibilità più complesse di combinazione sono spiegate in ⟨undefined⟩ [Simultaneous expressions], pagina ⟨undefined⟩):

```
\grace { g8( a b }
<> ) \p \< -. -\markup \italic "sempre staccato"
\repeat unfold 4 { c4 e }  c1\f
```



Si può usare la modalità relativa per indicare l'altezza degli accordi. La prima nota di ogni accordo è sempre relativa alla prima nota dell'accordo che lo precede oppure, se non c'è un accordo precedente, è relativa all'altezza dell'ultima nota che precede l'accordo. Le altezze di tutte le altre note dell'accordo sono relative alla nota che le precede *all'interno dell'accordo.*

```
<a c e>1 <f a c> <a c e> <f' a c> <b, e b,>
```



Maggiori informazioni sugli accordi si trovano in Sezione 2.7 [Chord notation], pagina 392.

Vedi anche

Glossario Musicale: Sezione “accordo” in *Glossario Musicale*.

Manuale d’apprendimento: Sezione “Combinare le note negli accordi” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 2.7 [Chord notation], pagina 392, ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩, ⟨undefined⟩ [Relative octave entry], pagina ⟨undefined⟩, ⟨undefined⟩ [Multiple voices], pagina ⟨undefined⟩.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

Problemi noti e avvertimenti

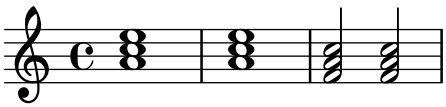
Gli accordi che contengono più di due altezze in uno spazio del rigo, come ad esempio ‘`<e f! fis!>`’, presentano le teste di tali note sovrapposte. A seconda della situazione, si può migliorare l’aspetto con

- l’uso temporaneo di ⟨undefined⟩ [Multiple voices], pagina ⟨undefined⟩, ‘`<< f! \\ <e fis!> >>`’,
- la trascrizione enarmonica di una o più altezze, ‘`<e f ges>`’, oppure
- i ⟨undefined⟩ [Clusters], pagina ⟨undefined⟩.

Ripetizione di un accordo

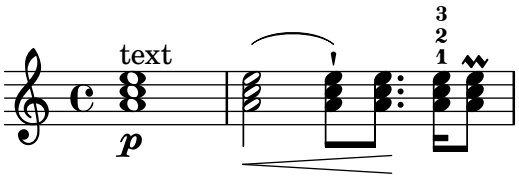
Per inserire la musica più rapidamente, si può usare una scorciatoia che ripete l’accordo precedente. Il simbolo di ripetizione dell’accordo è `q`:

```
<a c e>1 q <f a c>2 q
```



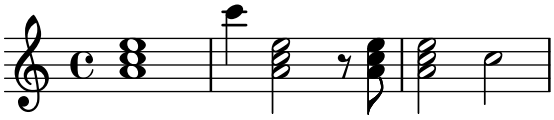
Come nel caso dei normali accordi, il simbolo di ripetzione dell’accordo si può usare con le durate, le articolazioni, i testi a margine, le legature di portamento, le travature, etc. dato che solo le altezze dell’accordo precedente vengono duplicate.

```
<a c e>1\p^"text" q2\<( q8)[-! q8.]\! q16-1-2-3 q8\prall
```
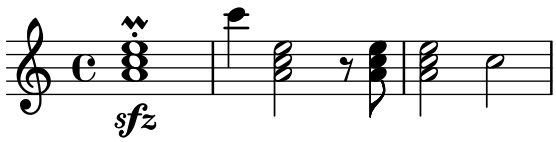


Il simbolo di ripetizione dell’accordo ricorda sempre l’ultimo accordo inserito, quindi è possibile inserire l’accordo più recente anche se nel frattempo sono state inserite altre note (senza accordi) o pause.

```
<a c e>1 c'4 q2 r8 q8 |
q2 c, |
```
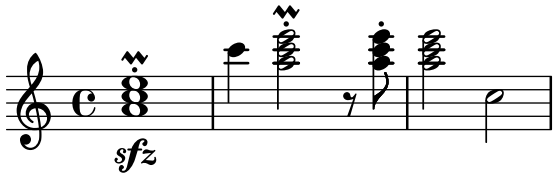


Tuttavia questo simbolo non conserva le dinamiche, le articolazioni o gli abbellimenti dell’accordo precedente.

```
<a-. c\prall e>1\sfz c'4 q2 r8 q8 |
q2 c, |
```



Per far sì che alcuni elementi siano conservati, si può invocare esplicitamente la funzione `\chordRepeats` con un'ulteriore argomento che indica una lista di *tipi di evento* da mantenere, a meno che eventi di quel tipo non siano già presenti nell'accordo `q` stesso.
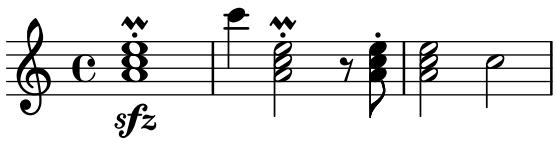
```
\relative c'' {
  \chordRepeats #'(articulation-event)
  { <a-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c, |
}
```



In questo esempio l'uso di `\chordRepeats` all'interno di un blocco `\relative` produce risultati indesiderati: gli eventi di un accordo, una volta espansi, non si distinguono più per essere stati inseriti come accordi normali, quindi `\relative` assegna un'ottava basata sul contesto corrente.

Dato che `\relative` annidati non si influenzano l'un l'altro, si può usare un altro `\relative` dentro `\chordRepeats` per stabilire le relazioni di ottava prima di espandere gli accordi ripetuti. In questo caso l'intero contenuto del `\relative` più interno non influenza quello esterno; ecco perché in questo esempio la nota finale è stata specificata con un'ottava diversa.

```
\new Voice
\relative c'' {
  \chordRepeats #'(articulation-event)
  \relative c''
  { <a-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c |
}
```



Le interazioni con `\relative` si verificano solo con chiamate esplicite di `\chordRepeats`: l'espansione implicita all'inizio della creazione della partitura viene fatta in un momento in cui tutti i `\relative` sono stati già elaborati.

Vedi anche

Guida alla notazione: Sezione 2.7 [Chord notation], pagina 392, ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.
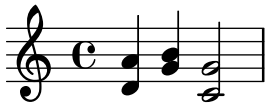
File installati: '`ly/chord-repetition-init.ly`'.

Espressioni simultanee
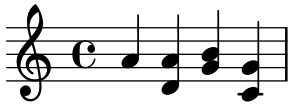
Una o più espressioni musicali racchiuse tra due coppie di parentesi uncinate sono considerate simultanee. Se la prima espressione inizia con una nota singola o se l'intera espressione simultanea appare esplicitamente all'interno di una voce, sarà posta in un solo rigo; altrimenti gli elementi dell'espressione simultanea saranno messi in righi separati.

Gli esempi seguenti mostrano espressioni simultanee su un rigo:

```
\new Voice {  % voce singola esplicita
  << { a4 b g2 } { d4 g c,2 } >>
}
```
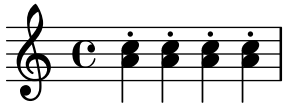


```
% prima nota singola
a << { a4 b g } { d4 g c, } >>
```



Questo può essere utile se le sezioni simultanee hanno durate identiche, ma i tentativi di collegare note con durate diverse allo stesso gambo causerà degli errori. Le note, le articolazioni e le modifiche delle proprietà in una *singola* voce ('`Voice`') sono raccolte e create secondo l'ordine della musica:
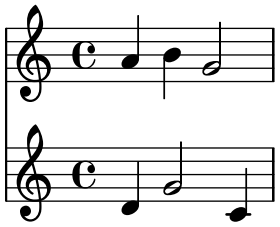
```
<a c>4-.  <>-. << c a >>  << { c-. <c a> } { a s-. } >>
```



Per poter inserire gambi o travature multiple e variare le durate o altre proprietà di note riferibili allo stesso momento musicale, occorre usare più voci.

L'esempio seguente mostra come le espressioni simultanee possano generare implicitamente righi multipli:

```
% nessuna singola nota precede l'espressione simultanea
<< { a4 b g2 } { d4 g2 c,4 } >>
```



In questo caso le durate diverse non causano problemi perché sono interpretate in voci diverse.

Problemi noti e avvertimenti

Se le note appartenenti a due o più voci, con gambi nella stessa direzione, si trovano nello stesso punto del rigo e non è stato specificato uno spostamento orizzontale (oppure è stato specificato lo stesso valore per lo spostamento), il messaggio:

```
attenzione: troppe collisioni tra colonne di note, ignorate
```

apparirà durante la compilazione del file. Si può evitare con:
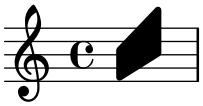
```
\override NoteColumn.ignore-collision = ##t
```

Tuttavia, questo comando non si limita a eliminare l'avvertimento, ma impedisce qualsiasi risoluzione delle collisioni, e potrebbe comportare altri effetti indesiderati (vedi anche i *Problemi noti* in ⟨undefined⟩ [Collision resolution], pagina ⟨undefined⟩).

Cluster

Un cluster prescrive l'esecuzione simultanea di tutti i suoni compresi in un determinato intervallo. Può essere rappresentato come un involucro che contiene le note che ne fanno parte. Si inserisce applicando la funzione `\makeClusters` a una sequenza di accordi, ad esempio:

```
\makeClusters { <g b>2 <c g'> }
```



Si possono inserire insieme sullo stesso rigo le normali note e i cluster, anche contemporaneamente. In tal caso non viene fatto alcun tentativo di evitare automaticamente collisioni tra le note normali e i cluster.

Vedi anche

Glossario Musicale: Sezione "cluster" in *Glossario Musicale*.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "ClusterSpanner" in *Guida al Funzionamento Interno*, Sezione "ClusterSpannerBeacon" in *Guida al Funzionamento Interno*, Sezione "Cluster_spanner_engraver" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

I cluster hanno un buon aspetto solo se durano almeno per due accordi; altrimenti appaiono troppo stretti.

I cluster non hanno un gambo e non possono indicare delle durate da soli, ma la lunghezza del cluster è determinata dalle durate degli accordi che lo definiscono. Più cluster distinti devono essere separati da una pausa.

I cluster non generano output MIDI.

1.5.2 Più voci
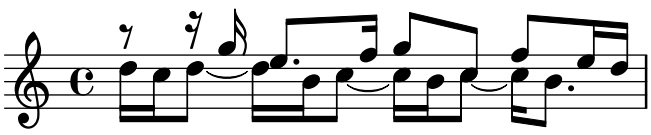
Questa sezione presenta le note simultanee in più voci o più righi.

Polifonia su un solo rigo

Istanziare esplicitamente le voci

La struttura di base necessaria per ottenere più voci indipendenti in un solo rigo è illustrata nell'esempio seguente:

```
\new Staff <<
  \new Voice = "prima"
    { \voiceOne r8 r16 g e8. f16 g8[ c,] f e16 d }
  \new Voice= "seconda"
    { \voiceTwo d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```



Le voci sono istanziate esplicitamente e vengono contrassegnate da dei nomi. I comandi `\voiceOne` ... `\voiceFour` impostano le voci in modo che la prima e terza voce abbiano i gambi in su, la seconda e la quarta voce i gambi in giù, le teste di nota della terza e quarta voce siano spostate orizzontalmente e le pause in ciascuna voce siano spostate automaticamente per evitare collisioni. Il comando `\oneVoice` ripristina tutte le impostazioni della voce alle direzioni neutrali predefinite.
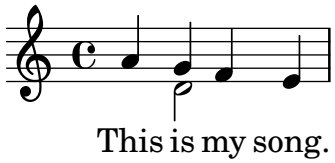
Pasaggi polifonici temporanei

Un passaggio polifonico temporaneo si può creare col seguente costrutto:

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

In questo esempio la prima espressione all'interno di un passaggio polifonico temporaneo è posta nel contesto `Voice` che era in uso immediatamente prima del passaggio polifonico e quello stesso contesto `Voice` continua dopo la sezione temporanea. Le altre espressioni comprese nelle parentesi uncinate vengono assegnate a voci temporanee distinte. Questo permette di assegnare il testo a una voce che continua prima, durante e dopo una sezione polifonica:

```
<<
  \new Voice = "melody" {
    a4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }
  \new Lyrics \lyricsto "melody" {
  This is my song.
  }
>>
```
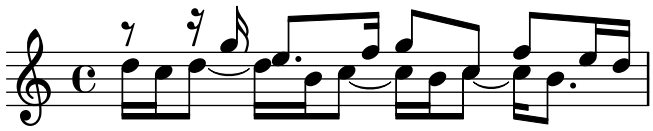


I comandi `\voiceOne` e `\voiceTwo` sono necessari per definire le impostazioni di ogni voce.

Il costrutto con la doppia barra inversa (backslash)

Il costrutto `<< {...} \\ {...} >>`, in cui due (o più) espressioni sono separate da due barre inverse (backslash), si comporta diversamente dal costrutto simile privo delle due barre: *tutte* le espressioni in questo costrutto vengono assegnate a nuovi contesti `Voice`. Questi nuovi contesti `Voice` vengono creati implicitamente e ad essi vengono assegnati dei nomi prestabiliti `"1"`, `"2"`, etc.

Il primo esempio potrebbe essere riscritto nel modo seguente:
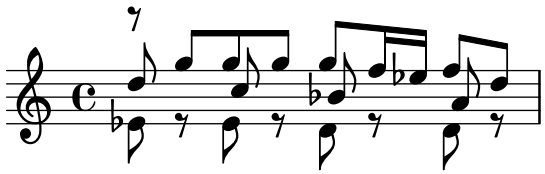
```
<<
  { r8 r16 g e8. f16 g8[ c,] f e16 d }
  \\
  { d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```



Questa sintassi si usa quando non importa che le voci temporanee siano create e poi eliminate. A queste voci create implicitamente vengono assegnate le impostazioni equivalenti all'uso dei comandi `\voiceOne` ... `\voiceFour`, nell'ordine in cui appaiono nell'input.

Nell'esempio seguente, la voce intermedia ha i gambi in su, dunque viene inserita in terza posizione in modo che diventi la terza voce, che ha i gambi in su. Si usano le pause spaziatrici per evitare il raddoppio delle pause ordinarie.

```
<<
  { r8 g g  g g f16 ees f8 d }
  \\
  { ees,8 r ees r d r d r }
  \\
  { d'8 s c s bes s a s }
>>
```



In tutti i brani, a eccezione di quelli più semplici, è consigliabile creare contesti `Voice` espliciti come è spiegato in Sezione "Contesti e incisori" in *Manuale di Apprendimento* e Sezione "Definire esplicitamente le voci" in *Manuale di Apprendimento*.

Ordine delle voci
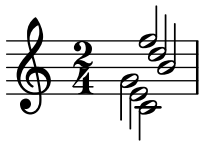
Quando si inseriscono più voci nel file di input, conviene usare il seguente ordine:

```
Voce 1: la più alta
Voce 2: la più bassa
Voce 3: la seconda più alta
```

```
Voce 4: la seconda più bassa
Voce 5: la  terza più alta
Voce 6: la terza più bassa
etc.
```

Sebbene possa sembrare controintuitivo, ciò semplifica il processo automatico di formattazione. Si noti che le voci con numero dispari hanno i gambi in su, quelle con numero pari hanno i gambi in giù:

```
\new Staff <<
  \time 2/4
  { f''2 }  % 1: la più alta
  \\
  { c'2  }  % 2: la più bassa
  \\
  { d''2 }  % 3: seconda più alta
  \\
  { e'2  }  % 4: seconda più bassa
  \\
  { b'2  }  % 5: terza più alta
  \\
  { g'2  }  % 6: terza più bassa
>>
```
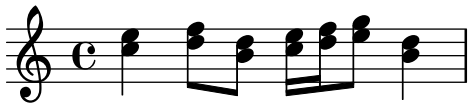


Nota: Il testo e gli estensori (come le legature di portamento e di valore, le forcelle, etc.) non possono 'attraversare' le voci.

Durate identiche

Nel caso speciale in cui si desideri inserire sezioni musicali parallele con il medesimo ritmo, si possono combinare in un unico contesto `Voice`, formando dunque degli accordi. Per farlo, vanno racchiuse in un semplice costrutto musicale simultaneo all'interno di una voce esplicita:

```
\new Voice <<
  { e4 f8 d e16 f g8 d4 }
  { c4 d8 b c16 d e8 b4 }
>>
```



Questo metodo produce strane travature e avvertimenti se le sezioni musicali non hanno lo stesso ritmo.

Comandi predefiniti

`\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`.

Vedi anche

Manuale d'apprendimento: Sezione "Le voci contengono la musica" in *Manuale di Apprendimento*, Sezione "Definire esplicitamente le voci" in *Manuale di Apprendimento*.

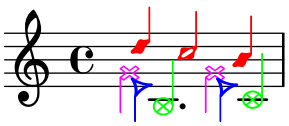
Guida alla notazione: [Percussion staves], pagina 372, ⟨undefined⟩ [Invisible rests], pagina ⟨undefined⟩, ⟨undefined⟩ [Stems], pagina ⟨undefined⟩.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Stili di voce

Si possono assegnare colori e forme diverse a ciascuna voce per facilitarne l'identificazione:

```
<<
  { \voiceOneStyle d4 c2 b4 }
  \\
  { \voiceTwoStyle e,2 e }
  \\
  { \voiceThreeStyle b2. c4 }
  \\
  { \voiceFourStyle g'2 g }
>>
```



Il comando `\voiceNeutralStyle` permette di ripristinare l'aspetto predefinito.

Comandi predefiniti

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`, `\voiceNeutralStyle`.

Vedi anche

Manuale d'apprendimento: Sezione "Sento le Voci" in *Manuale di Apprendimento*, Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.
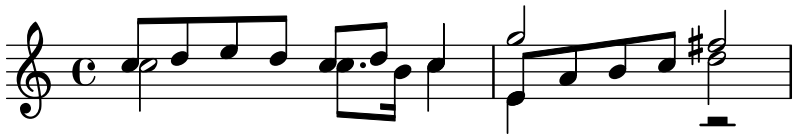
Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Risoluzione delle collisioni

Le teste di note che si trovano in voci diverse ma hanno stessa altezza, stessa testa e direzione del gambo opposta vengono unite automaticamente; invece, le note che hanno la stessa testa o la stessa direzione del gambo non vengono unite. Le pause opposte a un gambo in una voce diversa vengono spostate verticalmente. L'esempio seguente mostra tre diverse circostanze, sul primo e terzo movimento della prima battuta e sul primo movimento della seconda battuta, in cui l'unione automatica delle teste di nota non funziona.
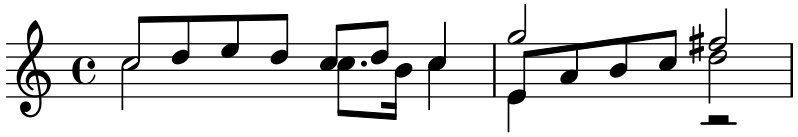
```
<<
  {
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
```

```
    s1
    e8 a b c d2
  }
>>
```



Note con teste diverse possono essere unite come è mostrato sotto. In questo esempio le teste delle note nel primo battito della prima battuta sono unite:
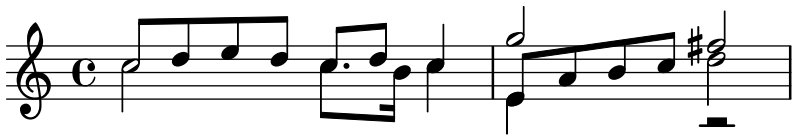
```
<<
  {
    \mergeDifferentlyHeadedOn
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



Le minime e le semiminime, invece, non sono unite, perché sarebbe difficile distinguerle.
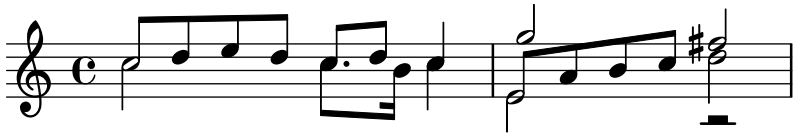
Anche le teste di note con diversi punti, come nel terzo movimento della prima battuta, possono essere unite:

```
<<
  {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



La minima e la croma all'inizio della seconda misura sono unite per errore, perché l'unione automatica non riesce a completare correttamente l'unione quando tre o più note sono allineate sulla stessa colonna di note: in questo caso la testa di nota unita non è corretta. Per far sì che l'unione selezioni la testa di nota corretta, occorre applicare il comando `\shift` alla nota che non deve essere unita. In questo esempio si usa `\shiftOn` per spostare il Sol superiore (g) fuori dalla colonna e di conseguenza `\mergeDifferentlyHeadedOn` funziona correttamente.

```
<<
  {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c8 d e d c d c4
    \shiftOn
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



Il comando `\shiftOn` permette (senza forzare) lo spostamento delle note in una voce. Quando si applica `\shiftOn` a una voce, una nota o accordo in quella voce vengono spostati solo se il suo gambo altrimenti entrerebbe in collisione col gambo di un'altra voce, e solo se i gambi che collidono puntano nella stessa direzione. Il comando `\shiftOff` impedisce che si verifichi questo tipo di spostamento.
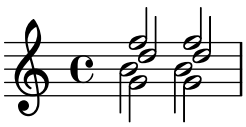
Per impostazione predefinita, le voci più esterne (solitamente la prima e la seconda voce) hanno specificato `\shiftOff`, mentre le voci più interne (terza e successive) hanno specificato `\shiftOn`. Quando si applica uno spostamento, le voci con i gambi in su (voci dispari) vengono spostate a destra, e le voci con i gambi in giù (voci pari) vengono spostate a sinistra.

Ecco un esempio che aiuta a visualizzare come un'espressione simultanea abbreviata viene espansa internamente.

> **Nota:** Attenzione: con tre o più voci, l'ordine verticale delle voci nel file di input non deve essere lo stesso dell'ordine verticale delle voci del rigo!

```
\new Staff \relative c'' {
  %% inserimento abbreviato
  <<
    { f2  }  % 1: più alta
    \\
    { g,2 }  % 2: più bassa
```

```
    \\
    { d'2 }  % 3: più alta centrale
    \\
    { b2  }  % 4: più  bassa centrale
  >>
  %% espansione interna dell'input precedente
  <<
    \new Voice = "1" { \voiceOne   \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo   \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn  d'2 } % sposta a destra
    \new Voice = "4" { \voiceFour  \shiftOn  b2  } % sposta a sinistra
  >>
}
```



Due ulteriori comandi, `\shiftOnn` e `\shiftOnnn`, mettono a disposizione altri livelli di spostamento che possono essere specificati in modo temporaneo per risolvere delle collisioni in situazioni complesse – vedi Sezione “Esempio musicale” in *Manuale di Apprendimento*.

Le note vengono unite solo se presentano opposta direzione dei gambi (come accade, ad esempio, nella prima o seconda voce o quando i gambi sono impostati esplicitamente in direzioni opposte).

Comandi predefiniti

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`.

Frammenti di codice selezionati
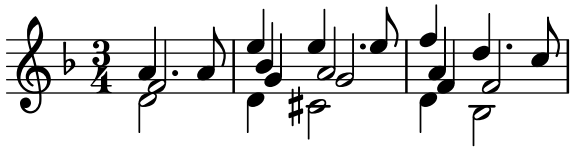
Voci ulteriori per evitare le collisioni

In alcuni casi di musica polifonica complessa sono necessarie delle voci ulteriori per evitare le collisioni tra note. Se servono più di quattro voci parallele, si possono aggiungere altre voci definendo una variabile con la funzione Scheme function `context-spec-music`.

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)

\relative c'' {
  \time 3/4
  \key d \minor
  \partial 2
  <<
    \new Voice  {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \new Voice {
      \voiceTwo
```
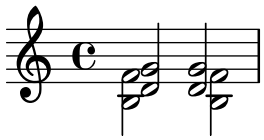
```
      d,2
      d4 cis2
      d4 bes2
    }
    \new Voice {
      \voiceThree
      f'2
      bes4 a2
      a4 s2
    }
    \new Voice {
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
  >>
}
```



Forzare lo spostamento orizzonatale delle note

Quando il motore tipografico non riesce a risolvere una situazione, si può usare la sintassi che sovrascrive le decisioni tipografiche. L'unità di misura usata è lo spazio del rigo.

```
\relative c' <<
  {
    <d g>2 <d g>
  }
  \\
  {
    <b f'>2
    \once \override NoteColumn.force-hshift = #1.7
    <b f'>2
  }
>>
```



Vedi anche

Glossario Musicale: Sezione "polifonia" in *Glossario Musicale*.

Manuale d'apprendimento: Sezione "Note simultanee" in *Manuale di Apprendimento*, Sezione "Le voci contengono la musica" in *Manuale di Apprendimento*, Sezione "Esempio musicale" in *Manuale di Apprendimento*.

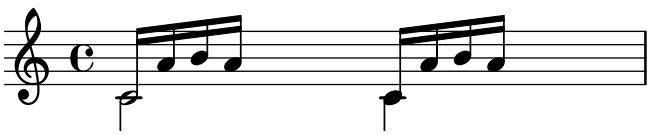
Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “NoteColumn” in *Guida al Funzionamento Interno*, Sezione “NoteCollision” in *Guida al Funzionamento Interno*, Sezione “RestCollision” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Se si usa `\override NoteColumn.ignore-collision = ##t`, le note con teste diverse che si trovano in voci diverse saranno unite in modo non corretto.

```
\mergeDifferentlyHeadedOn
<< { c16 a' b a } \\ { c,2 } >>
\override NoteColumn.ignore-collision = ##t
<< { c16 a' b a } \\ { c,2 } >>
```



Combinazione automatica delle parti

La combinazione automatica delle parti si usa per combinare in un unico rigo due parti musicali separate. Ciò è utile soprattutto quando si scrivono partiture orchestrali. Viene stampata una sola voce se le due parti musicali sono identiche, ma nei punti in cui sono diverse viene aggiunta una seconda voce. Le direzioni dei gambi sono impostate in su e in giù in base alla voce di appartenenza, mentre le sezioni solistiche e *a due* sono a loro volta identificate e contrassegnate.

La sintassi per la combinazione automatica delle parti è:

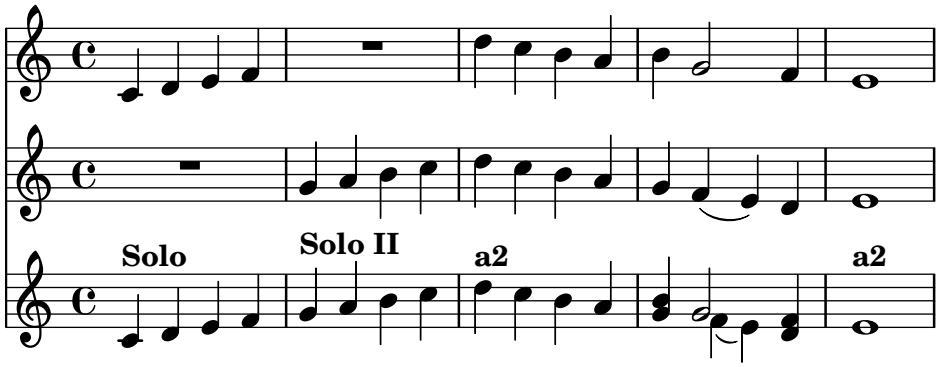
```
\partcombine espressione-musicale1 espressione-musicale2
```

L’esempio seguente illustra il funzionamento di base: le parti sono poste su un unico rigo in modo polifonico e le direzioni dei gambi sono regolate di conseguenza. Si usano le stesse variabili per le parti indipendenti e il rigo combinato.

```
instrumentOne = \relative c' {
  c4 d e f |
  R1 |
  d'4 c b a |
  b4 g2 f4 |
  e1 |
}

instrumentTwo = \relative g' {
  R1 |
  g4 a b c |
  d4 c b a |
  g4 f( e) d |
  e1 |
}

<<
  \new Staff \instrumentOne
  \new Staff \instrumentTwo
  \new Staff \partcombine \instrumentOne \instrumentTwo
>>
```



Entrambe le parti hanno note identiche nella terza misura, dunque viene stampata una sola nota. Le direzioni dei gambi e delle legature di portamento e di valore sono impostate automaticamente, a seconda che l'esecuzione delle parti sia solistica o all'unisono. Quando si rende necessario, in caso di polifonia, la prima parte (nel contesto `one`) ha i gambi in "su", mentre la seconda (nel contesto `two`) ha sempre i gambi in "giù". In caso di parti solistiche, la prima e seconda parte sono contrassegnate rispettivamente con "Solo" e "Solo II". Le parti (*a due*) all'unisono sono contrassegnate con la scritta "a2".

Entrambi gli argomenti di `\partcombine` sono interpretati come contesti `Voice` separati, dunque se la musica viene inserita in modo relativo *entrambe* le parti devono contenere una funzione `\relative`, ovvero:

```
\partcombine
  \relative ... espressione-musicale1
  \relative ... espressione-musicale2
```

Un blocco `\relative` che racchiude un `\partcombine` non ha effetto sulle altezze di *espressione-musicale1* e *espressione-musicale2*.

Nelle partiture professionali, spesso le voci sono tenute separate per lunghi passaggi anche se alcune note sono le stesse in entrambe le voci e potrebbero essere stampate come unisono. Combinare le note in un accordo o mostrare una voce come solista, dunque, non è la scelta ideale, perché la funzione `\partcombine` considera ogni nota individualmente. In questo caso si può sovrascrivere la funzione `\partcombine` con i comandi elencati sotto.

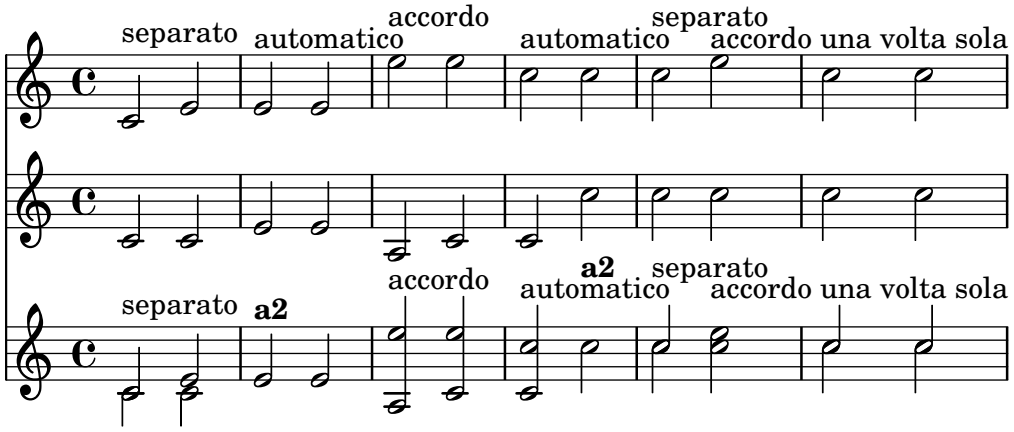
I comandi che finiscono con `...Once` si applicano soltanto alla nota successiva dell'espressione musicale.

- `\partcombineApart` e `\partcombineApartOnce` mantengono le note su due voci distinte, anche se potrebbero essere combinate in un accordo o in un unisono.
- `\partcombineChords` e `\partcombineChordsOnce` uniscono le note in un accordo.
- `\partcombineUnisono` e `\partcombineUnisonoOnce` uniscono entrambe le voci come "unisono".
- `\partcombineSoloI` e `\partcombineSoloIOnce` stampano soltanto la prima voce e la contrassegnano come un "Solo".
- `\partcombineSoloII` o `\partcombineSoloIIOnce` stampano soltanto la seconda voce e la contrassegnano come un "Solo".
- `\partcombineAutomatic` e `\partcombineAutomaticOnce` terminano le funzioni dei comandi precedenti e ripristinano il funzionamento predefinito di `\partcombine`.

```
instrumentOne = \relative c' {
  \partcombineApart c2^"separato" e |
  \partcombineAutomatic e2^"automatico" e |
  \partcombineChords e'2^"accordo" e |
  \partcombineAutomatic c2^"automatico" c |
  \partcombineApart c2^"separato" \partcombineChordsOnce e^"accordo una volta sola" |
  c2 c |
```

```
}
instrumentTwo = \relative c' {
  c2 c |
  e2 e |
  a,2 c |
  c2 c' |
  c2 c |
  c2 c |
}

<<
  \new Staff { \instrumentOne }
  \new Staff { \instrumentTwo }
  \new Staff { \partcombine \instrumentOne \instrumentTwo }
>>
```



Uso di \partcombine col testo vocale

Il comando `\partcombine` non è progettato per funzionare col testo vocale; al punto che se una delle voci è nominata in modo esplicito per poterle assegnare del testo, l'unione delle parti smette di funzionare. Tuttavia, questo risultato si può ottenere usando un contesto `NullVoice`. Vedi [Polyphony with shared lyrics], pagina 276.

Frammenti di codice selezionati

Combinare due parti sullo stesso rigo

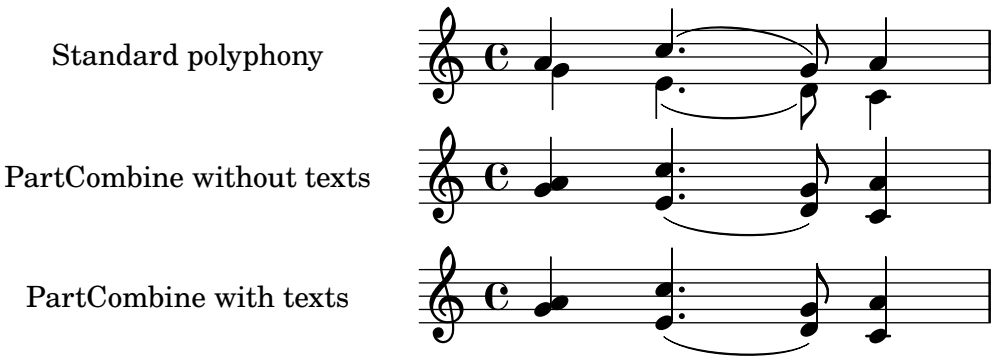
Lo strumento di unione delle parti (il comando `\partcombine`) permette di combinare varie parti sullo stesso rigo. Indicazioni testuali come "solo" e "a2" sono aggiunte automaticamente; per toglierele basta impostare la proprietà `printPartCombineTexts` su `f`. Per le partiture vocali (inni), non c'è bisogno di aggiungere i testi "solo/a2", quindi dovrebbero essere disattivati. Tuttavia potrebbe convenire non usarlo se c'è una qualche parte solista, perché non verrebbe indicata. In tali casi è preferibile usare la notazione polifonica normale.
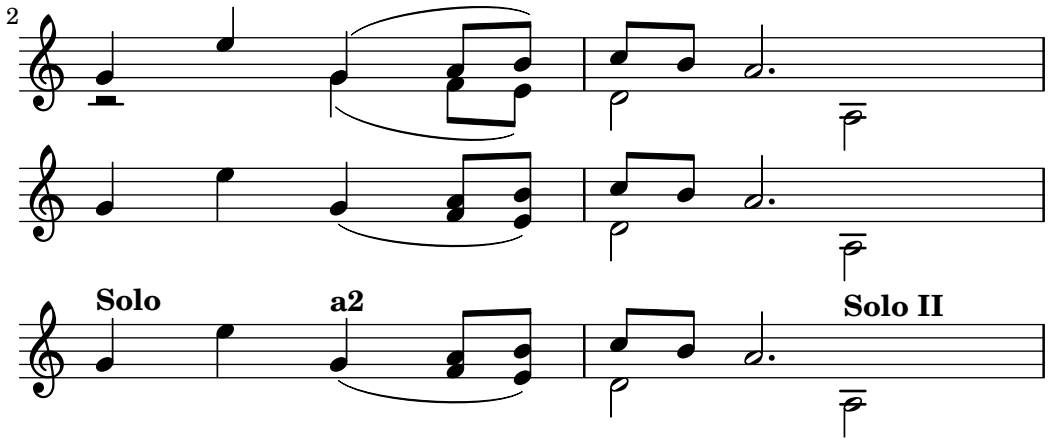
Questo frammento illustra i tre modi con cui due parti possono essere stampate su uno stesso rigo: normale polifonia, `\partcombine` senza testo e `\partcombine` con testo.

```
musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
  c b a2.
}
```

```
musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}

\score {
  <<
    <<
    \new Staff {
      \set Staff.instrumentName = #"Standard polyphony"
      << \musicUp \\ \musicDown >>
    }
    \new Staff \with { printPartCombineTexts = ##f } {
      \set Staff.instrumentName = #"PartCombine without texts"
      \partcombine \musicUp \musicDown
    }
    \new Staff {
      \set Staff.instrumentName = #"PartCombine with texts"
      \partcombine \musicUp \musicDown
    }
    >>
  >>
  \layout {
    indent = 6.0\cm
    \context {
      \Score
      \override SystemStartBar.collapse-height = #30
    }
  }
}
```
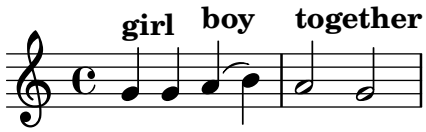




Modificare le indicazioni testuali di partcombine

Quando si usa la funzionalità di combinazione automatica delle parti, si può modificare il testo delle sezioni soliste e dell'unisono:

```
\new Staff <<
  \set Staff.soloText = #"girl"
  \set Staff.soloIIText = #"boy"
  \set Staff.aDueText = #"together"
  \partcombine
    \relative c'' {
      g4 g r r
      a2 g
    }
    \relative c'' {
      r4 r a( b)
      a2 g
    }
>>
```



Vedi anche

Glossario Musicale: Sezione "a due" in *Glossario Musicale*, Sezione "parte" in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Writing parts], pagina ⟨undefined⟩.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "PartCombineMusic" in *Guida al Funzionamento Interno*, Sezione "Voice" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Tutte le funzioni `\partcombine...` possono accettare soltanto due voci.

Le funzioni `\partcombine...` non possono essere inserite all'interno di un blocco `\tuplet` o `\relative`.

Se `printPartCombineTexts` è attivo e le due voci eseguono le stesse note "in modo discontinuo" nella stessa misura, potrebbe apparire il testo `a2` più di una volta in quella misura.

`\partcombine` sa soltanto quando una nota inizia in una voce (`Voice`); non può, ad esempio, ricordare se una nota in una voce è già iniziata quando combina le note già iniziate nell'altra

voce. Questo può portare a vari problemi inattesi, tra cui la stampa non corretta dei segni "Solo" e "Unisono".

`\partcombine` tiene tutti gli estensori (legature di portamento e di valore, forcelle, etc.) nella stessa voce, quindi se uno di questi estensori inizia o termina in una voce diversa potrebbe essere stampato incorrettamente o non essere stampato affatto.

Se la funzione `\partcombine` non riesce a combinare le due espressioni musicali (ovvero quando le due voci hanno durate diverse), assegnerà alle voci, internamente, nomi personalizzati: rispettivamente `one` e `two`. Ciò significa che se c'è un "passaggio" a un contesto `Voice` nominato diversamente, gli eventi in quel contesto verranno ignorati.

Consultare i *Problemi noti e avvertimenti* in [Default tablatures], pagina 326 se si usa `\partcombine` con l'intavolatura, e la *Nota* in ⟨undefined⟩ [Automatic beams], pagina ⟨undefined⟩ se si usa la disposizione automatica delle travature.

Scrivere la musica in parallelo

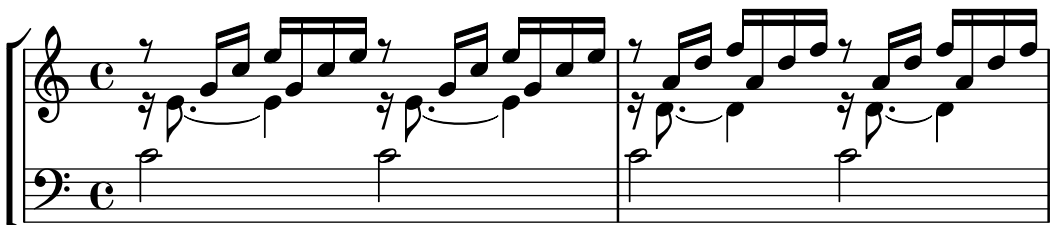
La musica che contiene parti diverse può essere messa in parallelo nel codice di input. La funzione `\parallelMusic` accetta una lista contenente i nomi di un insieme di variabili da creare e un'espressione musicale. Il contenuto delle misure alternate nell'espressione diventa il valore delle rispettive variabili, in modo che possano essere usate successivamente per stampare la musica.

> **Nota:** L'uso dei controlli di battuta `|` è obbligatorio e le misure devono avere la stessa durata.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Battuta 1
  r8 g'16 c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~   e'4           r16 e'8.~   e'4           |
  c'2                       c'2                       |

  % Battuta 2
  r8 a'16 d'' f'' a' d'' f'' r8 a'16 d'' f'' a' d'' f'' |
  r16 d'8.~   d'4           r16 d'8.~   d'4           |
  c'2                       c'2                       |

}
\new StaffGroup <<
  \new Staff << \voiceA \\ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
```
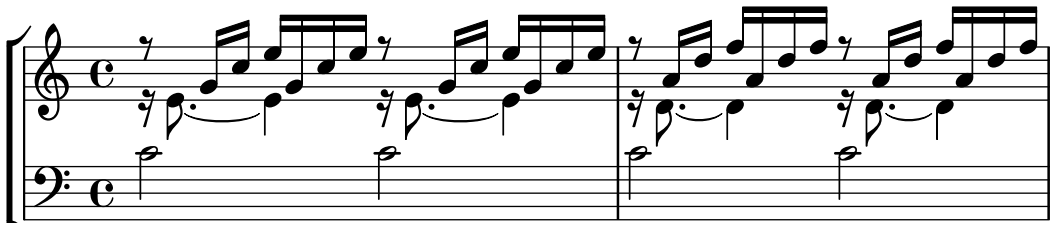


L'uso del modo relativo è permesso. Attenzione: il comando `\relative` non deve essere messo dentro `\parallelMusic`. Le note sono relative alla nota precedente della voce, non a quella precedente nell'input. In altre parole, le note relative di `voiceA` ignorano le note in `voiceB`.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Battuta 1
  r8 g16 c e g, c e r8 g,16 c e g, c e  |
  r16 e8.~ e4       r16 e8.~  e4        |
  c2                c                   |

  % Battuta 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~  d4       r16 d8.~  d4       |
  c2                 c                  |

 }
\new StaffGroup <<
  \new Staff << \relative c'' \voiceA \\ \relative c' \voiceB >>
  \new Staff \relative c' { \clef bass \voiceC }
>>
```



Questo è molto utile nella musica per pianoforte. Questo esempio combina sezioni di quattro battute consecutive con quattro variabili:

```
global = {
  \key g \major
  \time 2/4
}
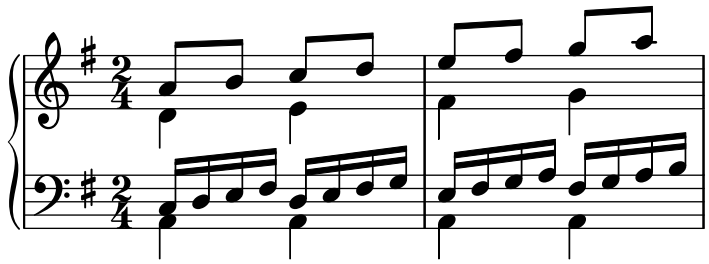
\parallelMusic #'(voiceA voiceB voiceC voiceD) {
  % Battuta 1
  a8    b     c   d     |
  d4          e         |
  c16 d e fis d e fis g |
  a4          a         |

  % Battuta 2
  e8      fis  g     a   |
  fis4         g         |
  e16 fis g  a fis g a b |
  a4           a         |

  % Bar 3 ...
}

\score {
  \new PianoStaff <<
     \new Staff {
       \global
       <<
```

```
        \relative c'' \voiceA
        \\
        \relative c'  \voiceB
      >>
    }
    \new Staff {
      \global \clef bass
      <<
        \relative c \voiceC
        \\
        \relative c \voiceD
      >>
    }
  >>
}
```

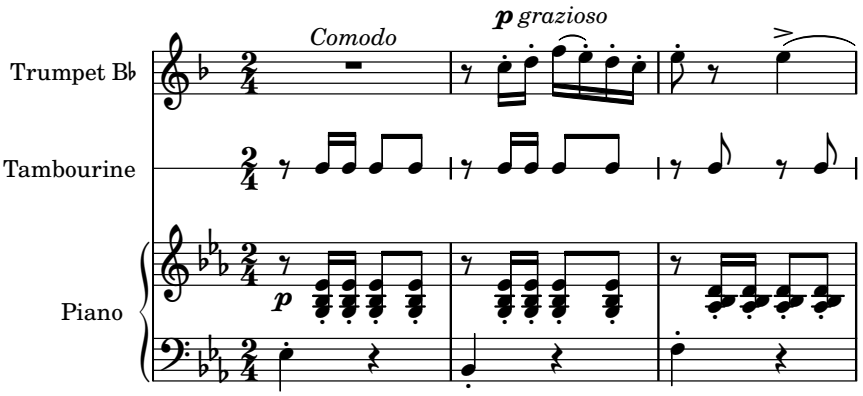


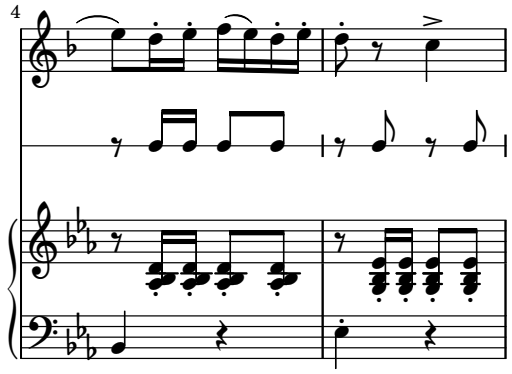
Vedi anche

Manuale d'apprendimento: Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

1.6 Notazione del rigo





Questa sezione spiega come modificare l'aspetto del rigo, come stampare partiture multirigo e come aggiungere indicazioni di tempo e citazioni in corpo più piccolo nel rigo.

1.6.1 Aspetto del rigo
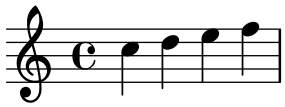
Questa sezione presenta i diversi metodi per creare e raggruppare i righi.

Istanziare nuovi righi

Il *rigo musicale* si crea con i comandi `\new` o `\context`. Ulteriori dettagli in Sezione 5.1.2 [Creating and referencing contexts], pagina 555.

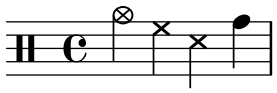
Il contesto di base del rigo è `Staff`:

```
\new Staff { c4 d e f }
```



Il contesto `DrumStaff` crea un rigo di cinque linee impostato per una tipica batteria. Ogni strumento viene mostrato con un simbolo diverso. Gli strumenti si inseriscono nella modalità percussioni, che si attiva col comando `\drummode`: ogni strumento viene indicato con un nome. Ulteriori dettagli in [Percussion staves], pagina 372.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



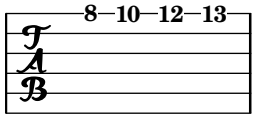
`RhythmicStaff` crea un rigo con una sola linea che mostra soltanto i valori ritmici dell'input. Le durate reali vengono mantenute. Ulteriori dettagli in ⟨undefined⟩ [Showing melody rhythms], pagina ⟨undefined⟩.

```
\new RhythmicStaff { c4 d e f }
```



`TabStaff` crea un'intavolatura (o tablatura) con sei corde nell'accordatura standard per chitarra. Ulteriori dettagli in [Default tablatures], pagina 326.
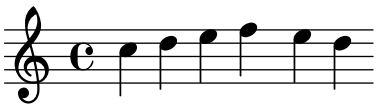
```
\new TabStaff { c4 d e f }
```



Ci sono due contesti del rigo specifici per la notazione di musica antica, `MensuralStaff` e `VaticanaStaff`, descritti in [Pre-defined contexts], pagina 415.

Il contesto `GregorianTranscriptionStaff` crea un rigo per il canto gregoriano moderno. Non mostra le stanghette delle battute.

```
\new GregorianTranscriptionStaff { c4 d e f e d }
```



Si possono creare nuovi contesti per un singolo rigo, come è spiegato dettagliatamente in Sezione 5.1.6 [Defining new contexts], pagina 568.

Vedi anche

Glossario musicale: Sezione “rigo” in *Glossario Musicale*,

Guida alla notazione: Sezione 5.1.2 [Creating and referencing contexts], pagina 555, [Percussion staves], pagina 372, ⟨undefined⟩ [Showing melody rhythms], pagina ⟨undefined⟩, [Default tablatures], pagina 326, [Pre-defined contexts], pagina 415, ⟨undefined⟩ [Staff symbol], pagina ⟨undefined⟩, [Gregorian chant contexts], pagina 424, [Mensural contexts], pagina 417, Sezione 5.1.6 [Defining new contexts], pagina 568.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

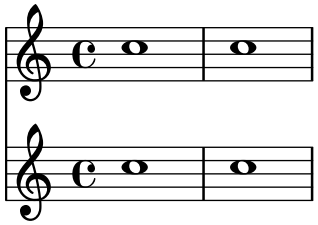
Guida al funzionamento interno: Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “DrumStaff” in *Guida al Funzionamento Interno*, Sezione “GregorianTranscriptionStaff” in *Guida al Funzionamento Interno*, Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*, Sezione “TabStaff” in *Guida al Funzionamento Interno*, Sezione “MensuralStaff” in *Guida al Funzionamento Interno*, Sezione “VaticanaStaff” in *Guida al Funzionamento Interno*, Sezione “StaffSymbol” in *Guida al Funzionamento Interno*.

Raggruppare i righi

Esistono vari contesti per raggruppare insieme singoli righi in modo da formare sistemi multirigo. Ogni contesto di raggruppamento imposta il comportamento delle stanghette e lo stile del segno che delimita l’inizio del sistema.

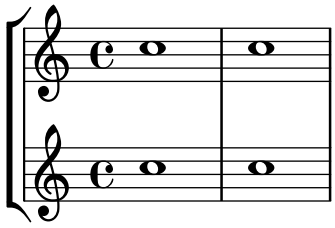
Se non si specifica alcun contesto, vengono usate le proprietà predefinite: il gruppo inizia con una linea verticale e le stanghette non sono collegate.

```
<<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



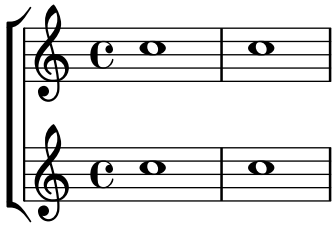
Nel contesto `StaffGroup`, il gruppo inizia con una parentesi quadra e le stanghette attraversano tutti i righi.

```
\new StaffGroup <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



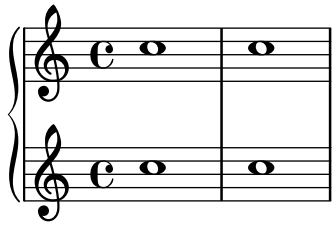
Nel contesto `ChoirStaff`, il gruppo inizia con una parentesi quadra, ma le stanghette non sono collegate.

```
\new ChoirStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



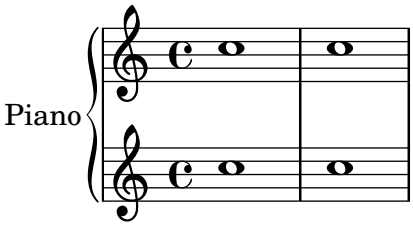
Nel contesto `GrandStaff`, il gruppo inizia con una parentesi graffa e le stanghette sono collegate da rigo a rigo.

```
\new GrandStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



Il contesto `PianoStaff` è identico a `GrandStaff`, con l'unica differenza che permette di mostrare il nome dello strumento direttamente. Ulteriori dettagli in ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩.

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano"
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



Ogni contesto per il gruppo di righi imposta la proprietà `systemStartDelimiter` su uno dei seguenti valori: `SystemStartBar`, `SystemStartBrace` o `SystemStartBracket`. È presente anche un quarto delimitatore, `SystemStartSquare`, ma deve essere indicato esplicitamente.

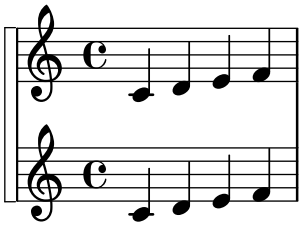
Si possono definire nuovi contesti di gruppi di rigo. I dettagli sono spiegati in Sezione 5.1.6 [Defining new contexts], pagina 568.

Frammenti di codice selezionati

Usare una parentesi quadra all'inizio di un gruppo di righi

Si può usare il segno `SystemStartSquare` (uno dei segni che delimitano l'inizio del sistema) impostandolo esplicitamente in un contesto `StaffGroup` o `ChoirStaff`.

```
\score {
  \new StaffGroup { <<
  \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```
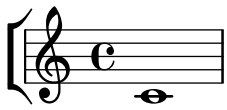


Mostrare la parentesi anche se c'è un solo rigo nel sistema

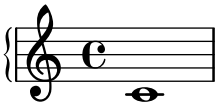
Se c'è un solo rigo in uno dei tipi di rigo `ChoirStaff` o `StaffGroup`, la parentesi e la stanghetta iniziale non appaiono. Si può modificare questo comportamento predefinito sovrascrivendo `collapse-height` e impostando un valore inferiore al numero di linee del rigo.

Nei contesti `PianoStaff` e `GrandStaff`, dove i sistemi iniziano con una parentesi graffa invece di una parentesi quadra, occorre impostare un'altra proprietà, come si vede nel secondo sistema dell'esempio.

```
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
```

```
    \new Staff {
      c'1
    }
  >>
}
```

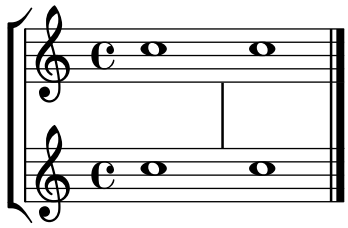




Formattazione mensurale (stanghette tra i righi)

La formattazione mensurale, in cui le stanghette non appaiono sui righi ma nello spazio tra i righi, si può ottenere usando `StaffGroup` al posto di `ChoirStaff`. La stanghetta sui righi viene nascosta impostando la proprietà `transparent`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Vedi anche

Glossario musicale: Sezione “graffa” in *Glossario Musicale*, Sezione “parentesi quadra” in *Glossario Musicale*, Sezione “accollatura” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩, Sezione 5.1.6 [Defining new contexts], pagina 568.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “StaffGroup” in *Guida al Funzionamento Interno*, Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “GrandStaff” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “SystemStartBar” in *Guida al Funzionamento*

Interno, Sezione “SystemStartBrace” in *Guida al Funzionamento Interno*, Sezione “SystemStartBracket” in *Guida al Funzionamento Interno*, Sezione “SystemStartSquare” in *Guida al Funzionamento Interno*.
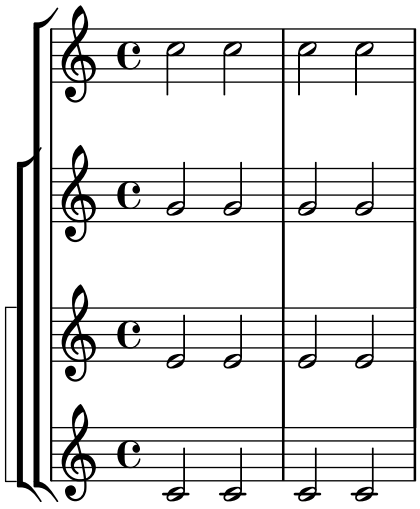
Problemi noti e avvertimenti

`PianoStaff` non accetta `ChordNames`.

Gruppi di righi annidati

I contesti dei gruppi di righi possono essere annidati fino a qualsiasi livello. In questo caso, ogni contesto inferiore crea una nuova parentesi accanto alla parentesi del gruppo superiore.

```
\new StaffGroup <<
  \new Staff { c2 c | c2 c }
  \new StaffGroup <<
    \new Staff { g2 g | g2 g }
    \new StaffGroup \with {
      systemStartDelimiter = #'SystemStartSquare
    }
    <<
      \new Staff { e2 e | e2 e }
      \new Staff { c2 c | c2 c }
    >>
  >>
>>
```



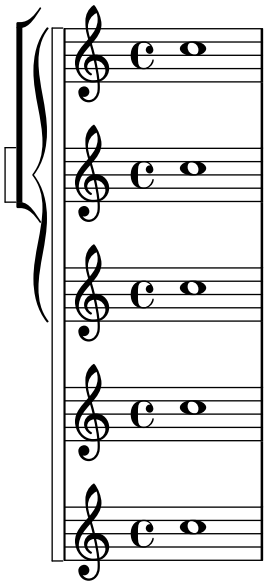
Si possono definire nuovi gruppi di righi annidati. Ulteriori dettagli in Sezione 5.1.6 [Defining new contexts], pagina 568.

Frammenti di codice selezionati

Annidare i righi

Si può usare la proprietà `systemStartDelimiterHierarchy` per creare gruppi di righi annidati più complessi. Il comando `\set StaffGroup.systemStartDelimiterHierarchy` prende come argomento una lista alfabetica dell’insieme di righi prodotti. Prima di ogni rigo si può assegnare un delimitatore di inizio del sistema. Deve essere racchiuso tra parentesi e collega tutti i righi compresi tra le parentesi. Gli elementi nella lista possono essere omessi, ma la prima parentesi quadra collega sempre tutti i righi. Le possibilità sono `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace` e `SystemStartSquare`.

```
\new StaffGroup
\relative c'' <<
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
                              (SystemStartSquare b)  ) c ) d)
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩, Sezione 5.1.6 [Defining new contexts], pagina 568.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

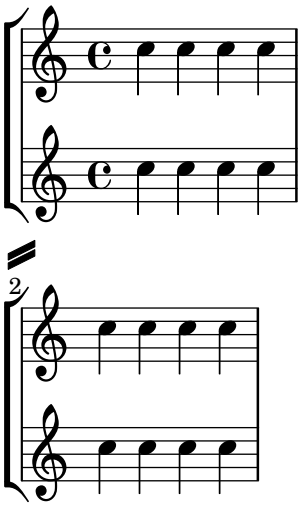
Guida al funzionamento interno: Sezione “StaffGroup” in *Guida al Funzionamento Interno*, Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “SystemStartBar” in *Guida al Funzionamento Interno*, Sezione “SystemStartBrace” in *Guida al Funzionamento Interno*, Sezione “SystemStartBracket” in *Guida al Funzionamento Interno*, Sezione “SystemStartSquare” in *Guida al Funzionamento Interno*.

Separare i sistemi

Se il numero di sistemi per pagina cambia di pagina in pagina, è consuetudine separare i sistemi con un segno separatore. Per impostazione predefinita questo segno è disattivo, ma può essere attivato con un’opzione in `\paper`.

```
\book {
  \score {
    \new StaffGroup <<
      \new Staff {
        \relative c'' {
          c4 c c c
          \break
```

```
            c4 c c c
          }
        }
        \new Staff {
          \relative c'' {
            c4 c c c
            \break
            c4 c c c
          }
        }
      >>
    }
    \paper {
      system-separator-markup = \slashSeparator
      % i seguenti comandi servono soltanto alla formattazione di questa documentazione
      paper-width = 100\mm
      paper-height = 100\mm
      tagline = ##f
    }
  }
```



Vedi anche

Guida alla notazione: Sezione 4.1 [Page layout], pagina 502.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

1.6.2 Modificare singoli righi
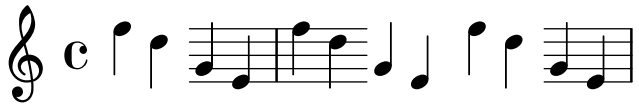
Questa sezione spiega come modificare gli attributi specifici di un rigo, per esempio il numero di linee o la dimensione del rigo. Vengono descritti anche i metodi per iniziare e finire un rigo e per impostare le sezioni ossia.

Simbolo del rigo

I comandi `\stopStaff` e `\startStaff` servono a fermare o (ri)avviare le linee del rigo, per impedire che appaiano in un punto della partitura.

```
\stopStaff f4 d \startStaff g, e
f'4 d \stopStaff g, e
```

```
f'4 d \startStaff g, e
```



Comandi predefiniti
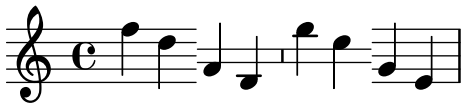
`\startStaff`, `\stopStaff`.

Le linee di un rigo appartengono all'oggetto `StaffSymbol` (che comprende i tagli addizionali) e si possono modificare tramite le proprietà di `StaffSymbol`; però queste modifiche devono essere fatte prima che il rigo sia (ri)avviato.

Si può cambiare il numero di linee del rigo:

```
f4 d \stopStaff
\override Staff.StaffSymbol.line-count = #2
\startStaff g, e |

f'4 d \stopStaff
\revert Staff.StaffSymbol.line-count
\startStaff g, e |
```



Si può cambiare anche la posizione di ogni linea del rigo. Un elenco di numeri definisce la posizione di ogni linea. I valori consueti sono `0` per la linea centrale e `(-4 -2 0 2 4)` per le altre. La linea del rigo appare solo se è presente il suo valore, quindi questo comando permette di variare anche il numero delle linee, oltre alla loro posizione.

```
f4 d \stopStaff
\override Staff.StaffSymbol.line-positions = #'(1 3 5 -1 -3)
\startStaff g, e |
f'4 d \stopStaff
\override Staff.StaffSymbol.line-positions = #'(8 6.5 -6 -8 -0.5)
\startStaff g, e
```
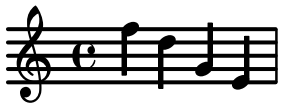


Per conservare le tipiche direzioni dei gambi (nella metà inferiore del rigo i gambi puntano in su, mentre in quella superiore sono rivolti in giù), occorre allineare la linea centrale (o lo spazio) del rigo personalizzato alla posizione della linea centrale normale (0). Potrà essere necessario regolare la posizione della chiave e del Do centrale per adattarsi alle nuove linee. Si veda ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

Si può modificare lo spessore della linea del rigo. Per impostazione predefinita, questa modifica ha effetto anche sui tagli addizionali e sui gambi.
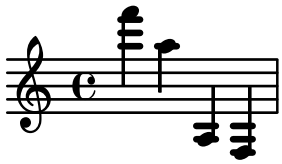
```
\new Staff \with {
  \override StaffSymbol.thickness = #3
}
```

```
{ f4 d g, e }
```



È anche possibile impostare lo spessore dei tagli addizionali in modo indipendente dalle linee del rigo.

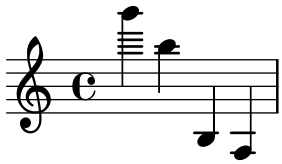
```
\new Staff \with {
  \override StaffSymbol.thickness = #2
  \override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
}
{ f'4 a, a,, f }
```



Il primo valore viene moltiplicato per lo spessore della linea del rigo, il secondo per la spaziatura del rigo; la somma dei due valori definisce il nuovo valore dello spessore del taglio addizionale.
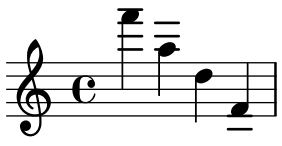
Si possono modificare le posizioni verticali dei tagli addizionali:

```
\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
}
{ f'4 a, a,, f }
```
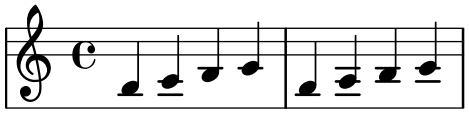


Si possono far apparire ulteriori tagli addizionali sopra o sotto le teste delle note, a seconda della posizione corrente relativa alle altre teste, anch'esse con i propri tagli addizionali.

```
\new Staff \with {
  \override StaffSymbol.ledger-extra = #4
}
{ f'4 a, d, f, }
```



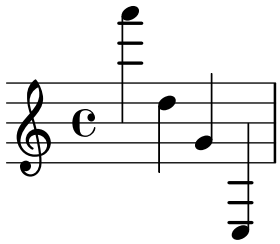
Si possono far apparire i tagli addizionali anche dentro il rigo quando servono delle linee personalizzate. L'esempio mostra la posizione predefinita dei tagli addizionali quando la proprietà `ledger-position` è impostata e quando non lo è. Nell'esempio il comando `\stopStaff` serve ad annullare il comando `\override` per l'oggetto `StaffSymbol`.

```
\override Staff.StaffSymbol.line-positions = #'(-8 0 2 4)
d4 e f g
\stopStaff
\startStaff
\override Staff.StaffSymbol.ledger-positions = #'(-8 -6 (-4 -2) 0)
d4 e f g
```



Si può cambiare la distanza tra le linee del rigo. Tale modifica ha effetto anche sulla spaziatura della linea.

```
\new Staff \with {
  \override StaffSymbol.staff-space = #1.5
}
{ f'4 d, g, e, }
```
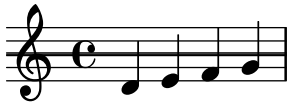


Frammenti di codice selezionati

Rendere alcune linee del rigo più spesse delle altre

In ambito didattico può essere utile rendere più spesso una linea del rigo (per esempio, la linea centrale, o per sottolineare la linea della chiave di Sol). Per farlo si possono aggiungere altre linee e posizionarle molto vicino alla linea che deve essere evidenziata, usando la proprietà `line-positions` dell'oggetto `StaffSymbol`.

```
{
  \override Staff.StaffSymbol.line-positions =
    #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



Vedi anche

Glossario musicale: Sezione "linea" in *Glossario Musicale*, Sezione "taglio addizionale" in *Glossario Musicale*, Sezione "rigo (o pentagramma)" in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.
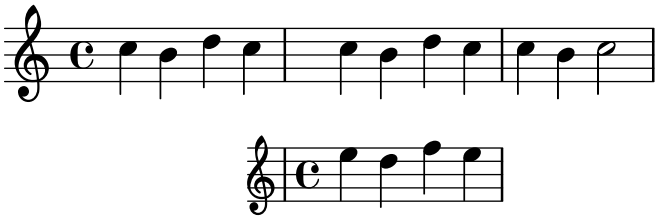
Frammenti: Sezione "Staff notation" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "StaffSymbol" in *Guida al Funzionamento Interno*, Sezione "staff-symbol-interface" in *Guida al Funzionamento Interno*.

Righi ossia

I righi *ossia* si possono creare aggiungendo un nuovo rigo simultaneo nel punto giusto:

```
\new Staff \relative c'' {
  c4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
  >>
  c4 b c2
}
```

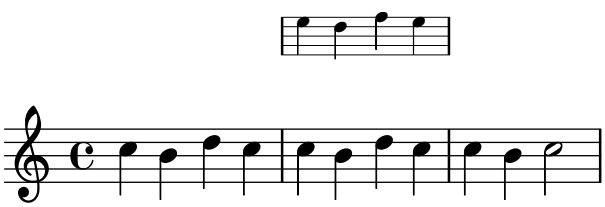


Tuttavia, questo esempio non produce quel che normalmente si desidera. Per creare righi ossia che siano sopra il rigo originale, non abbiano indicazione di tempo né chiave e abbiano un tipo di carattere più piccolo, sono necessarie delle modifiche manuali. Il Manuale d'apprendimento descrive una tecnica specifica per ottenere questo risultato, a partire da Sezione "Annidare le espressioni musicali" in *Manuale di Apprendimento*.

L'esempio seguente usa la proprietà `alignAboveContext` per allineare il rigo ossia. Questo metodo conviene quando sono necessari solo pochi righi ossia.

```
\new Staff = "main" \relative c'' {
  c4 b d c
  <<
    { c4 b d c }

    \new Staff \with {
      \remove "Time_signature_engraver"
      alignAboveContext = #"main"
      fontSize = #-3
      \override StaffSymbol.staff-space = #(magstep -3)
      \override StaffSymbol.thickness = #(magstep -3)
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```
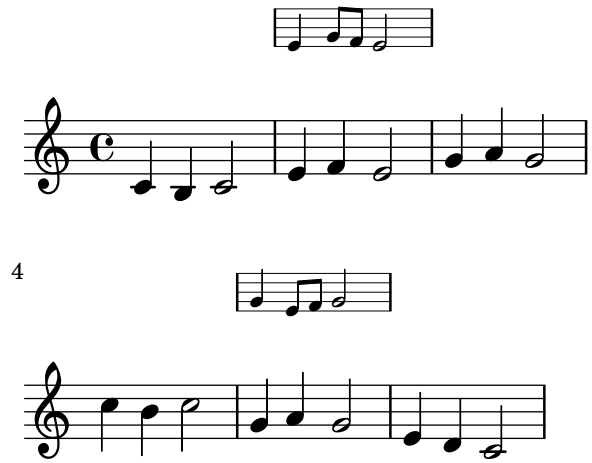


Se si hanno molti righi ossia isolati, è meglio creare un contesto `Staff` vuoto con un *identificativo del contesto* specifico; i righi ossia possono essere creati *chiamando* questo contesto e

usando `\startStaff` e `\stopStaff` nei punti richiesti. I vantaggi di questo metodo sono più evidenti se il brano è più lungo del seguente esempio.

```
<<
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
    \hide Clef
    fontSize = #-3
    \override StaffSymbol.staff-space = #(magstep -3)
    \override StaffSymbol.thickness = #(magstep -3)
  }
  { \stopStaff s1*6 }

  \new Staff \relative c' {
    c4 b c2
    <<
      { e4 f e2 }
      \context Staff = "ossia" {
        \startStaff e4 g8 f e2 \stopStaff
      }
    >>
    g4 a g2 \break
    c4 b c2
    <<
      { g4 a g2 }
      \context Staff = "ossia" {
        \startStaff g4 e8 f g2 \stopStaff
      }
    >>
    e4 d c2
  }
>>
```
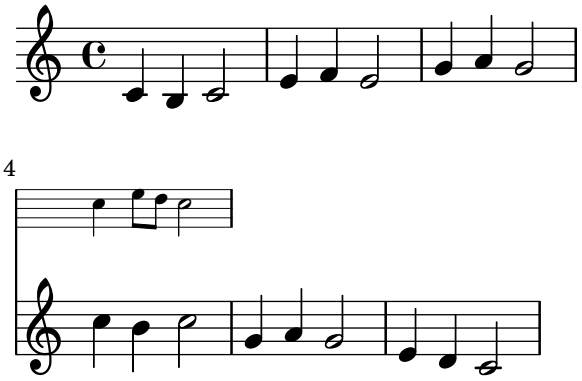


Come alternativa, si può usare il comando `\Staff \RemoveEmptyStaves` per creare i righi ossia. Questo metodo conviene quando i righi ossia si trovano subito dopo un'interruzione di linea. Ulteriori informazioni su `\Staff \RemoveEmptyStaves` si trovano in ⟨undefined⟩ [Hiding staves], pagina ⟨undefined⟩.

```
<<
```

```
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
    \hide Clef
    fontSize = #-3
    \override StaffSymbol.staff-space = #(magstep -3)
    \override StaffSymbol.thickness = #(magstep -3)
  } \relative c'' {
    R1*3
    c4 e8 d c2
  }
  \new Staff \relative c' {
    c4 b c2
    e4 f e2
    g4 a g2 \break
    c4 b c2
    g4 a g2
    e4 d c2
  }
>>

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    \override VerticalAxisGroup.remove-first = ##t
  }
}
```



Frammenti di codice selezionati

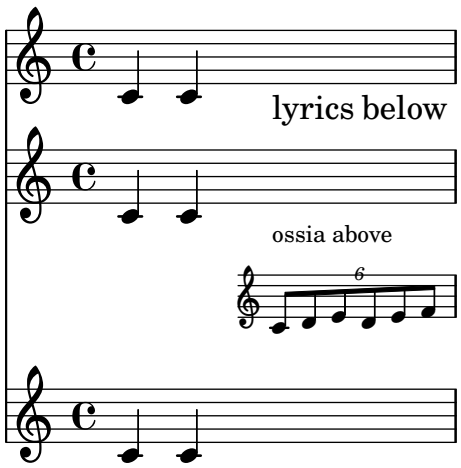
Allineare verticalmente gli ossia e il testo vocale

Questo frammento mostra come usare le proprietà di contesto `alignBelowContext` e `alignAboveContext` per controllare il posizionamento del testo vocale e degli ossia.

```
\paper {
  ragged-right = ##t
}

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
```

```
{ \skip 2
  <<
    \lyrics {
      \set alignBelowContext = #"1"
      lyrics4 below
    }
    \new Staff \with {
      alignAboveContext = #"3"
      fontSize = #-2
      \override StaffSymbol.staff-space = #(magstep -2)
      \remove "Time_signature_engraver"
    } {
      \tuplet 6/4 {
        \override TextScript.padding = #3
        c8[^"ossia above" d e d e f]
      }
    }
  >>
}
>>
```



Vedi anche

Glossario musicale: Sezione “ossia” in *Glossario Musicale*, Sezione “rigo (pentagramma)” in *Glossario Musicale*, Sezione “rigo temporaneo” in *Glossario Musicale*.

Manuale d’apprendimento: Sezione “Annidare le espressioni musicali” in *Manuale di Apprendimento*, Sezione “Dimensione degli oggetti” in *Manuale di Apprendimento*, Sezione “Lunghezza e spessore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Hiding staves], pagina ⟨undefined⟩.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “StaffSymbol” in *Guida al Funzionamento Interno*.

Nascondere i righi

Le linee del rigo si possono nascondere togliendo l’incisore `Staff_symbol_engraver` dal contesto `Staff`. Altrimenti si può usare `\stopStaff`.

```
\new Staff \with {
  \remove "Staff_symbol_engraver"
```

```
}
\relative c''' { a8 f e16 d c b a2 }
```

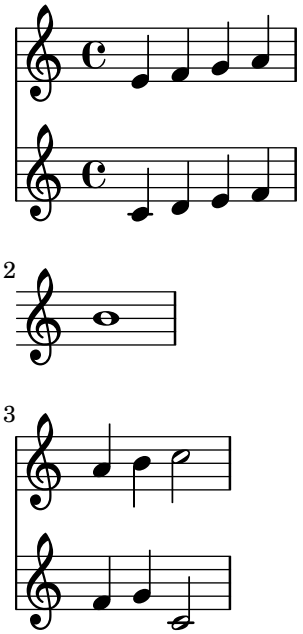


I righi vuoti si possono nascondere inserendo il comando `\Staff \RemoveEmptyStaves` nel blocco `\layout`. Nelle partiture per orchestra, questo stile è noto come 'Partitura alla francese'. Questo comando nasconde e toglie tutti i righi vuoti di una partitura eccetto quelli nel primo sistema.

> **Nota:** Un rigo viene considerato vuoto quando contiene soltanto pause multiple, pause, salti, pause spaziatrici o una combinazione di questi elementi.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}

\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
    a4 b c2
  }
  \new Staff {
    c,4 d e f \break
    R1 \break
    f4 g c,2
  }
>>
```



`\Staff \RemoveEmptyStaves` si può usare anche per creare sezioni ossia per un rigo. I dettagli si trovano in ⟨undefined⟩ [Ossia staves], pagina ⟨undefined⟩.

Per nascondere i righi vuoti nei contesti della musica antica si può usare il comando `\VaticanaStaff \RemoveEmptyStaves`. Analogalmente, `\RhythmicStaff \RemoveEmptyStaves` permette di nascondere i contesti `RhythmicStaff` vuoti.

Comandi predefiniti

`\Staff \RemoveEmptyStaves`, `\VaticanaStaff \RemoveEmptyStaves`, `\RhythmicStaff \RemoveEmptyStaves`.
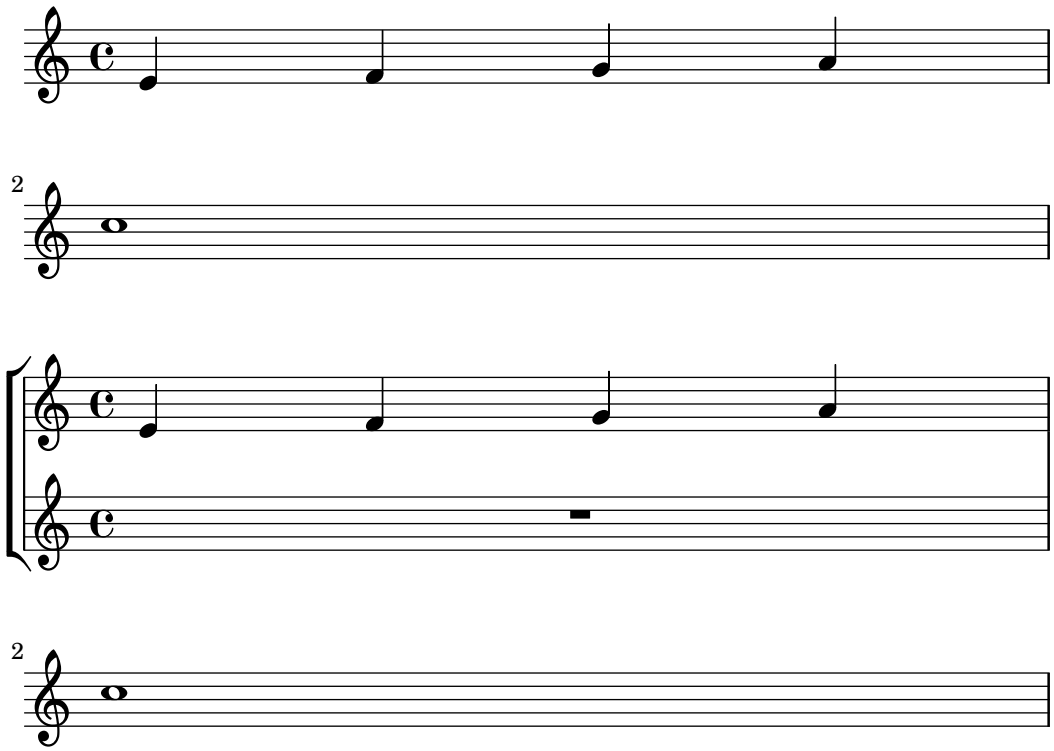
Frammenti di codice selezionati

Eliminare la prima linea vuota

Il primo rigo vuoto si può togliere dalla partitura impostando la proprietà `remove-first` di `VerticalAxisGroup`. Questa impostazione agisce a livello globale se posta nel blocco `\layout`, a livello locale se posta nel rigo specifico che deve essere tolto. Nel secondo caso, si deve specificare il contesto (`Staff` si applica solo al rigo corrente) prima della proprietà.

Il rigo inferiore del secondo gruppo di righi non viene rimosso, perché l'impostazione ha effetto solo sul rigo in cui si trova.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
    % To use the setting globally, uncomment the following line:
    % \override VerticalAxisGroup.remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup.remove-first = ##t
    R1 \break
    R
  }
>>
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    R1 \break
    R
  }
>>
```



Vedi anche

Glossario musicale: Sezione “rigo temporaneo” in *Glossario Musicale*.

Manuale d’apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.5 [Changing context default settings], pagina 563, ⟨undefined⟩ [Staff symbol], pagina ⟨undefined⟩, ⟨undefined⟩ [Ossia staves], pagina ⟨undefined⟩, ⟨undefined⟩ [Hidden notes], pagina ⟨undefined⟩, ⟨undefined⟩ [Invisible rests], pagina ⟨undefined⟩, Sezione 5.4.6 [Visibility of objects], pagina 592.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ChordNames” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “Staff_symbol_engraver” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Se si toglie l’incisore `Staff_symbol_engraver` vengono nascoste anche le stanghette. Se si forza la visibilità delle stanghette, potrebbero verificarsi degli errori di formattazione. In questo caso, conviene usare i seguenti comandi invece di togliere l’incisore:

```
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t
```

Per i problemi noti e gli avvertimenti relativi a `\Staff` `\RemoveEmptyStaves` si veda Sezione 5.1.5 [Changing context default settings], pagina 563.
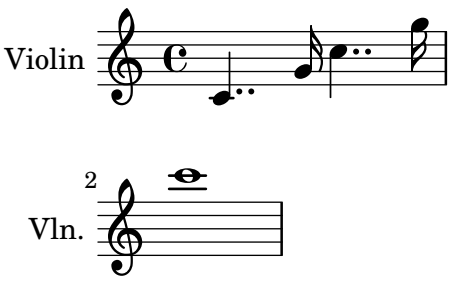
1.6.3 Scrittura delle parti

Questa sezione spiega come inserire in una partitura le indicazioni di tempo e i nomi degli strumenti. Mostra anche come citare altre voci e come formattare le citazioni in corpo più piccolo.

Nomi degli strumenti
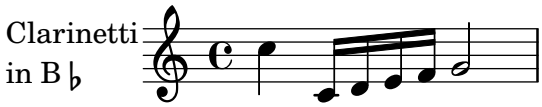
I nomi degli strumenti possono essere fatti apparire, alla sinistra dei righi, nei contesti `Staff`, `PianoStaff`, `StaffGroup`, `GrandStaff` e `ChoirStaff`. Il valore di `instrumentName` viene usato per il primo rigo e quello di `shortInstrumentName` per tutti i righi successivi.

```
\new Staff \with {
  instrumentName = #"Violin "
  shortInstrumentName = #"Vln. "
}
{ c4.. g'16 c4.. g'16 \break | c1 }
```
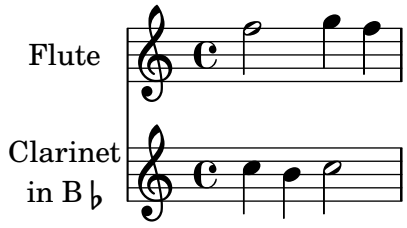


Si può usare `\markup` per creare nomi più complessi:

```
\new Staff \with {
  instrumentName = \markup {
    \column { "Clarinetti"
      \line { "in B" \smaller \flat }
    }
  }
}
{ c4 c,16 d e f g2 }
```



Se due o più contesti del rigo sono raggruppati insieme, i nomi degli strumenti, sia quello normale che quello abbreviato, vengono centrati automaticamente. Per allineare al centro i nomi degli strumenti che vanno a capo, occorre usare `\center-column`:

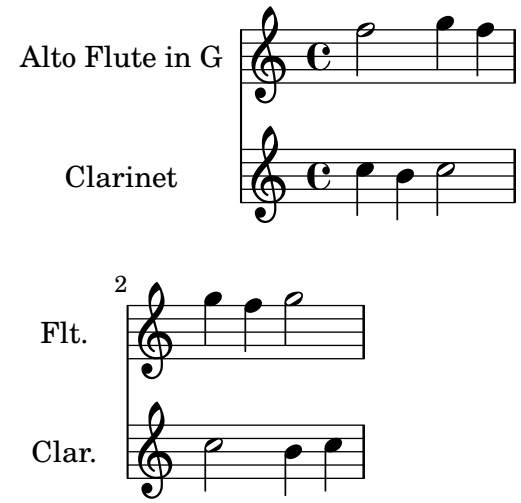
```
<<
  \new Staff \with {
    instrumentName = #"Flute"
  }
  { f2 g4 f }
  \new Staff \with {
    instrumentName = \markup {
      \center-column { "Clarinet"
        \line { "in B" \smaller \flat }
      }
    }
  }
  { c4 b c2 }
>>
```



Tuttavia, se i nomi degli strumenti sono lunghi, potranno essere centrati solo aumentando i valori di `indent` e `short-indent`. Ulteriori dettagli su queste impostazioni si trovano in [\paper variables for shifts and indents], pagina 509.

```
\relative c'' {
  <<
    \new Staff \with {
      instrumentName = #"Alto Flute in G"
      shortInstrumentName = #"Flt."
    }
    {
      f2 g4 f \break
      g4 f g2
    }
    \new Staff \with {
      instrumentName = #"Clarinet"
      shortInstrumentName = #"Clar."
    }
    {
      c,4 b c2 \break
      c2 b4 c
    }
  >>
}

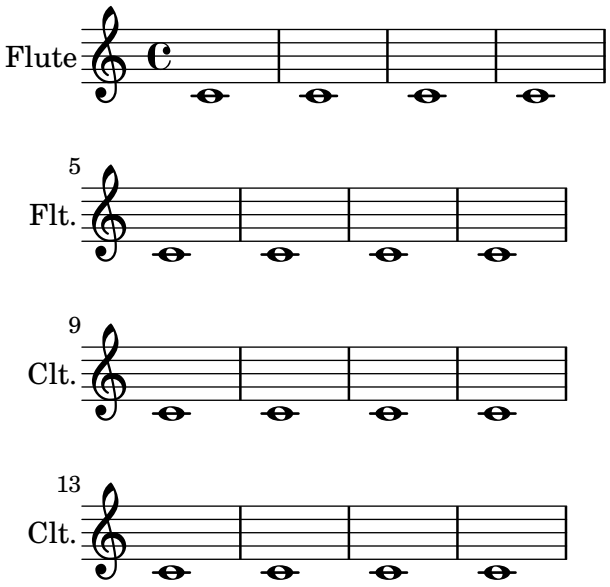
\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm
}
```



Per impostare i nomi degli strumenti in altri contesti (come `ChordNames` o `FiguredBass`), si deve aggiungere l'incisore `Instrument_name_engraver` a quel contesto. Ulteriori dettagli in Sezione 5.1.4 [Modifying context plug-ins], pagina 561.

`shortInstrumentName` può essere cambiato all'interno di un brano, mentre di `instrumentName` apparirà solo la prima definizione e le modifiche successive saranno ignorate:

```
\new Staff \with {
  instrumentName = #"Flute"
  shortInstrumentName = #"Flt."
}
{
  c1 c c c \break
  c1 c c c \break
  \set Staff.instrumentName = #"Clarinet"
  \set Staff.shortInstrumentName = #"Clt."
  c1 c c c \break
  c1 c c c \break
}
```

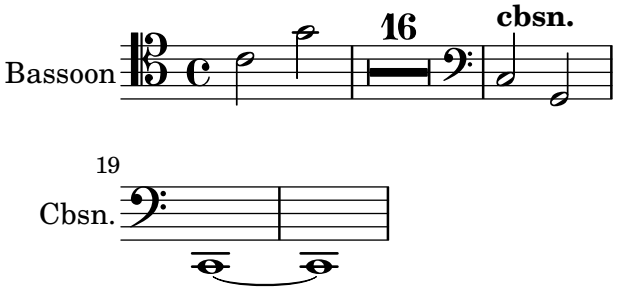


Se serve un *cambio* di strumento, si può usare `\addInstrumentDefinition` insieme a `\instrumentSwitch` per creare una lista dettagliata delle modifiche necessarie per il cambio. Il comando `\addInstrumentDefinition` prende due argomenti: una stringa testuale per identificare lo strumento, e una lista di associazione delle proprietà di contesto e dei valori da usare. Deve trovarsi nell'ambito di più alto livello. Per dichiarare il cambio di strumento, si usa il comando `\instrumentSwitch`, all'interno dell'espressione musicale, :

```
\addInstrumentDefinition #"contrabassoon"
  #`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
     (shortInstrumentName . "Cbsn.")
     (clefGlyph . "clefs.F")
     (middleCPosition . 6)
     (clefPosition . 2)
     (instrumentCueName . ,(make-bold-markup "cbsn."))
     (midiInstrument . "bassoon"))

\new Staff \with {
  instrumentName = #"Bassoon"
}
\relative c' {
```

```
  \clef tenor
  \compressFullBarRests
  c2 g'
  R1*16
  \instrumentSwitch "contrabassoon"
  c,,2 g \break
  c,1 ~ | c1
}
```



Vedi anche

Guida alla notazione: [\paper variables for shifts and indents], pagina 509, Sezione 5.1.4 [Modifying context plug-ins], pagina 561.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “InstrumentName” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

Citare altre voci

È molto comune che una voce usi le stesse note di un’altra voce. Per esempio, il primo e il secondo violino che suonano la stessa frase durante un particolare passaggio del brano. Per evitare di reinserire la musica di nuovo per la seconda voce, si può far sì che una voce *citi* l’altra.

Il comando `\addQuote`, usato nell’ambito di più alto livello, definisce un flusso musicale da cui poter citare i frammenti.

Il comando `\quoteDuring` serve a indicare il punto in cui inizia la citazione. È seguito da due argomenti: il nome della voce citata, come è definito da `\addQuote`, e un’espressione musicale per la durata della citazione.
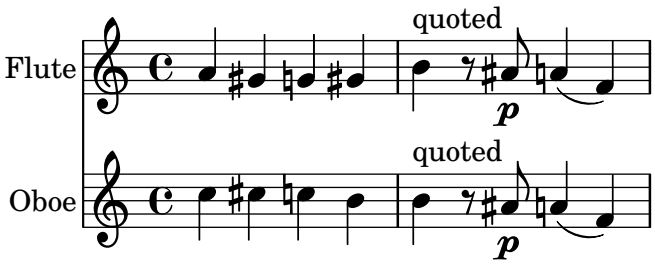
```
fluteNotes = \relative c'' {
  a4 gis g gis | b4^"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative c'' {
  c4 cis c b \quoteDuring #"flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
```
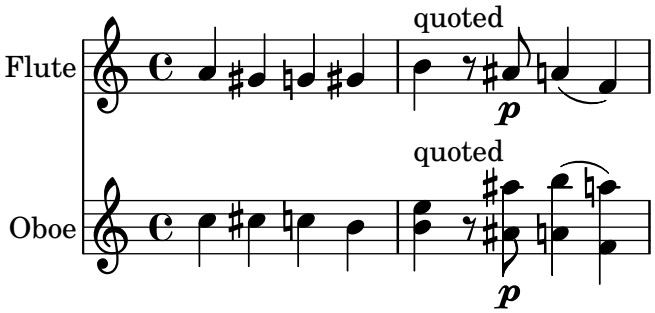
```
}
```



Se l'espressione musicale usata in `\quoteDuring` contiene note invece di pause spaziatrici o multiple, la citazione apparirà in forma polifonica e potrebbe causare risultati indesiderati.

```
fluteNotes = \relative c'' {
  a4 gis g gis | b4^"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative c'' {
  c4 cis c b \quoteDuring #"flute" { e4 r8 ais b4 a }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```
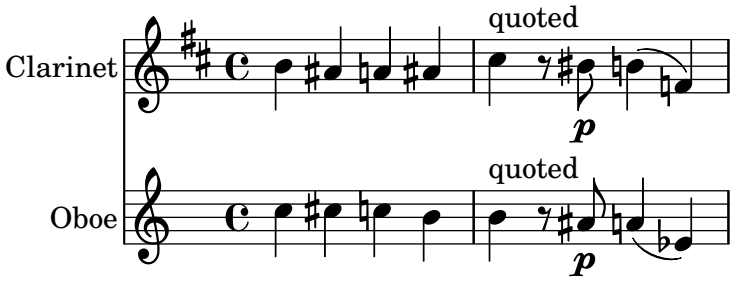


Il comando `\quoteDuring` usa le impostazioni `\transposition` sia della parte citata sia di quella che cita per produrre delle note per la parte che cita che abbiano la stessa altezza di quelle nella parte citata.

```
clarinetNotes = \relative c'' {
  \transposition bes
  \key d \major
  b4 ais a ais | cis4^"quoted" r8 bis\p b4( f)
}

oboeNotes = \relative c'' {
  c4 cis c b \quoteDuring #"clarinet" { s1 }
}
```

```
\addQuote "clarinet" { \clarinetNotes }


\score {
  <<
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```
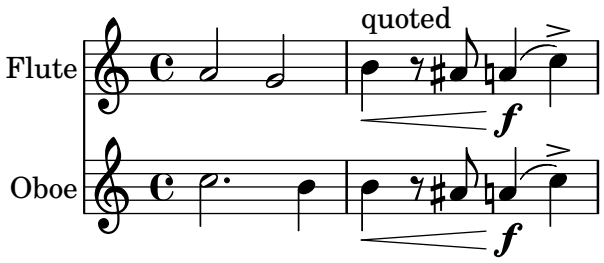


La musica citata include tutte le articolazioni, dinamiche, annotazioni, etc. presenti nel frammento citato. È possibile scegliere quali di questi oggetti far apparire usando la proprietà di contesto `quotedEventTypes`.

```
fluteNotes = \relative c'' {
  a2 g2 |
  b4\<^"quoted" r8 ais a4\f( c->)
 }

oboeNotes = \relative c'' {
  c2. b4 |
  \quoteDuring #"flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \set Score.quotedEventTypes = #'(note-event articulation-event
                                     crescendo-event rest-event
                                     slur-event dynamic-event)
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
 }
```



Le citazioni possono anche essere contrassegnate; si veda [Using tags], pagina 483.

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩, [Using tags], pagina 483.

File installati: ‘`scm/define-event-classes.scm`’.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Music classes” in *Guida al Funzionamento Interno*, Sezione “QuoteMusic” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.
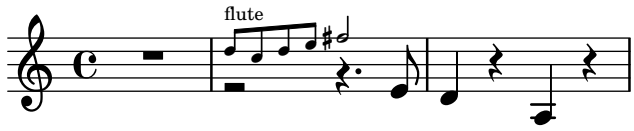
Problemi noti e avvertimenti

Solo il contenuto della prima voce in un comando `\addQuote` sarà preso in considerazione per la citazione; quindi se l’espressione musicale contiene comandi `\new` o `\context Voice`, il loro contenuto non verrà citato. La citazione degli abbellimenti non è supportata e potrebbe causare il crash di LilyPond; la citazione di terzine annidate potrebbe produrre una notazione mediocre.

Formattazione delle notine
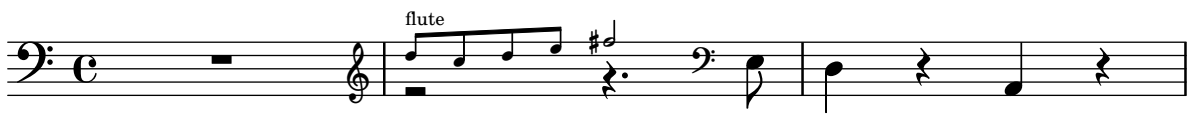
Il modo più semplice per formattare le notine è creare esplicitamente un contesto `CueVoice` all’interno della parte.

```
R1
<<
  { e2\rest r4. e8 }
  \new CueVoice {
    \stemUp d'8^"flute" c d e fis2
  }
>>
d,4 r a r
```
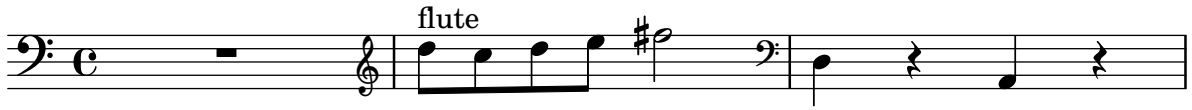


Si può usare il comando `\cueClef` all’interno di un contesto `CueVoice` esplicito se è richiesto un cambiamento di chiave; in questo modo la chiave apparirà nella dimensione giusta per le notine. Si può poi usare il comando `\cueClefUnset` per tornare alla chiave originale, di nuovo nella dimensione giusta.

```
\clef "bass"
R1
<<
  { e2\rest r4. \cueClefUnset e,8 }
  \new CueVoice {
    \cueClef "treble" \stemUp d''8^"flute" c d e fis2
  }
>>
d,,4 r a r
```



I comandi `\cueClef` e `\cueClefUnset` si possono usare anche senza un esplicito contesto `CueVoice`.

```
\clef "bass"
R1
\cueClef "treble"
d'8^"flute" c d e fis2
\cueClefUnset
d,,4 r a r
```



Per posizionamenti complessi delle notine, per esempio includere la trasposizione o inserire delle notine da varie sorgenti musicali, si possono usare i comandi `\cueDuring` o `\cueDuringWithClef`. Questi sono delle varianti più specializzate di `\quoteDuring`, introdotto in ⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩ nella sezione precedente.

La sintassi è:

```
\cueDuring #nomecitazione #direzione #musica
```

e

```
\cueDuringWithClef #nomecitazione #direzione #chiave #musica
```

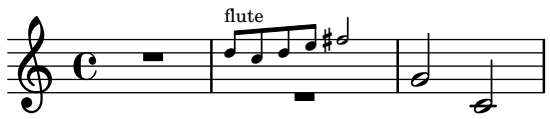
La musica delle misure che corrispondono a *`nomecitazione`* viene aggiunta in un contesto `CueVoice` e si colloca in simultanea con *`musica`*, creando quindi una situazione polifonica. La *`direzione`* prende l'argomento `UP` o `DOWN`, e corrisponde alla prima e alla seconda voce rispettivamente, determinando come le notine appaiono in relazione all'altra voce.

```
fluteNotes = \relative c'' {
  r2. c4 | d8 c d e fis2 | g2 d |
}

oboeNotes = \relative c'' {
  R1
  \new CueVoice { \set instrumentCueName = "flute" }
  \cueDuring #"flute" #UP { R1 }
  g2 c,
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \oboeNotes
}
```
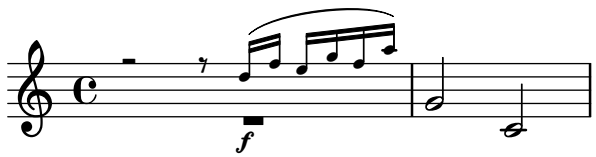


È possibile controllare quali aspetti della musica vengono citati con `\cueDuring` impostando la proprietà `quotedCueEventTypes`. Il suo valore predefinito è `'(note-event rest-event tie-event beam-event tuplet-span-event)`, che significa che vengono citati solo note, pause, legature di valore, travature e gruppi irregolari, ma non le articolazioni, le indicazioni dinamiche, il testo a margine, etc.

Nota: Quando una voce inizia con `\cueDuring`, come nell'esempio seguente, il contesto `Voice` deve essere dichiarato esplicitamente, altrimenti l'intera espressione musicale appartiene al contesto `CueVoice`.

```
oboeNotes = \relative c'' {
  r2 r8 d16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboeNotes }

\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                      beam-event tuplet-span-event
                                      dynamic-event slur-event)
  \cueDuring #"oboe" #UP { R1 }
  g2 c,
}
```



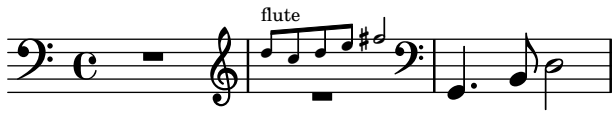
Il nome dello strumento che suona la citazione si imposta con la proprietà `instrumentCueName` in un contesto `CueVoice` temporaneo. Il posizionamento e lo stile di `instrumentCueName` è regolato dall'oggetto `instrumentSwitch`, vedi ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩. Se le citazioni in corpo più piccolo richiedono un cambio di chiave, si può fare manualmente ma anche il ripristino della chiave originale dovrà essere fatto manualmente al termine delle citazioni.

```
fluteNotes = \relative c'' {
  r2. c4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  \clef treble
  \new CueVoice { \set instrumentCueName = "flute" }
  \cueDuring #"flute" #UP { R1 }
  \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



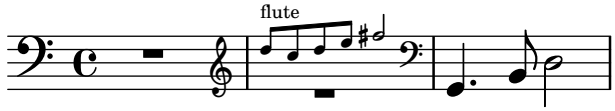
Altrimenti si può usare la funzione `\cueDuringWithClef`. Questo comando prende un ulteriore argomento per specificare il cambio di chiave da usare per le citazioni in corpo più piccolo ma mostrerà automaticamente la chiave originale appena le citazioni sono finite.

```
fluteNotes = \relative c'' {
  r2. c4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  \new CueVoice { \set instrumentCueName = "flute" }
  \cueDuringWithClef #"flute" #UP #"treble" { R1 }
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



Come `\quoteDuring`, `\cueDuring` prende in considerazione la trasposizione degli strumenti. Le citazioni in corpo più piccolo vengono mostrate nelle altezze necessarie allo strumento che riprende la citazione per riprodurre gli stessi suoni dello strumento citato.

Per trasporre le citazioni in corpo più piccolo in modo diverso, si usa `\transposedCueDuring`. Questo comando prende un ulteriore argomento per specificare (in modalità assoluta) l'altezza da usare nella partitura per rappresentare il Do centrale in intonazione reale. È utile nel caso di citazioni da uno strumento che ha un registro completamente diverso.

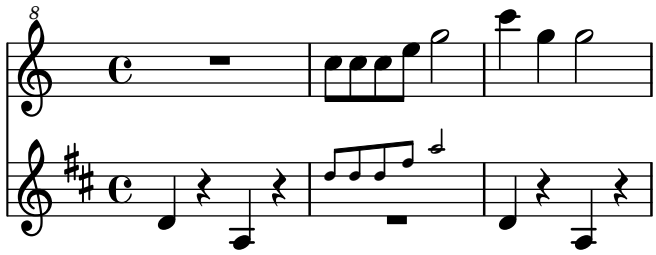
```
piccoloNotes = \relative c''' {
  \clef "treble^8"
  R1
  c8 c c e g2
  c4 g g2
}

bassClarinetNotes = \relative c' {
  \key d \major
  \transposition bes,
  d4 r a r
  \transposedCueDuring #"piccolo" #UP d { R1 }
  d4 r a r
}

\addQuote "piccolo" { \piccoloNotes }

<<
  \new Staff \piccoloNotes
```

```
  \new Staff \bassClarinetNotes
>>
```



Il comando `\killCues` toglie le citazioni in corpo più piccolo da un'espressione musicale, in modo che la stessa espressione musicale possa essere usata per produrre sia la parte strumentale con le citazioni in corpo più piccolo sia l'intera partitura. Il comando `\killCues` toglie soltanto le note e gli eventi citati da `\cueDuring`. Altre annotazioni relative alle citazioni in corpo più piccolo, come i cambi di chiave e il nome che identifica lo strumento sorgente, possono essere contrassegnate per includerle in modo selettivo nella partitura; si veda [Using tags], pagina 483.

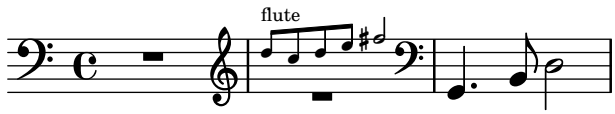
```
fluteNotes = \relative c'' {
  r2. c4 d8 c d e fis2 g2 d2
}

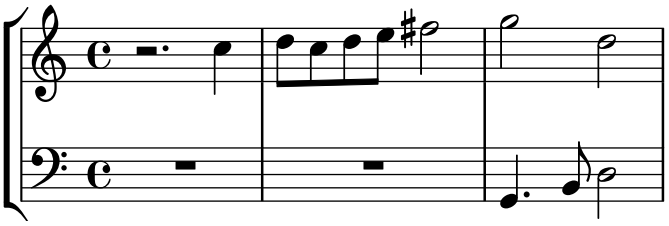
bassoonNotes = \relative c {
  \clef bass
  R1
  \tag #'part {
    \clef treble
    \new CueVoice { \set instrumentCueName = "flute" }
  }
  \cueDuring #"flute" #UP { R1 }
  \tag #'part \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

\new StaffGroup <<
  \new Staff {
    \fluteNotes
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoonNotes } }
  }
>>
```





Altrimenti, i cambi di chiave e i nomi identificativi degli strumenti possono essere inseriti in una definizione, in modo da poterli riutilizzare, col comando `\addInstrumentDefinition` descritto in ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩.

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩, ⟨undefined⟩ [Clef], pagina ⟨undefined⟩, [Musical cues], pagina 292, [Using tags], pagina 483.

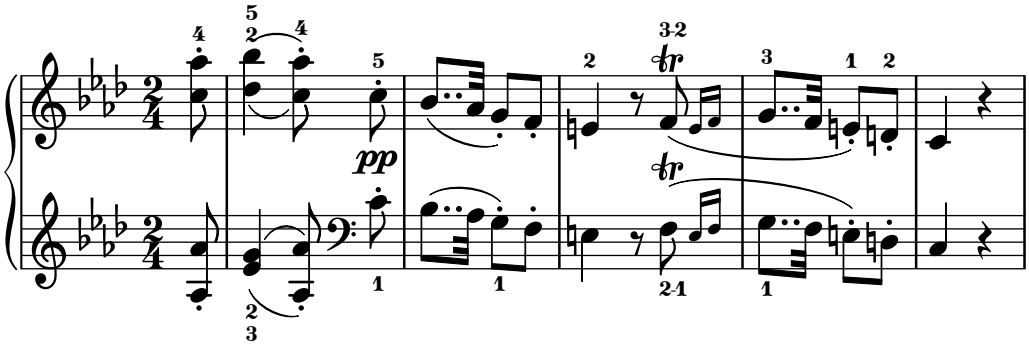
Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “CueVoice” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Quando si usa `\cueDuring`, si possono verificare delle collisioni tra le pause nel contesto `Voice` e quelle in `CueVoice`. Quando si usa `\cueDuringWithClef` o `\transposedCueDuring`, l’argomento ulteriore richiesto da entrambi deve venire dopo la citazione e la direzione.

1.7 Note editoriali



Questa sezione tratta dei vari modi con cui cambiare l’aspetto delle note e aggiungere un’analisi o un accento didattico.

1.7.1 Interne al rigo
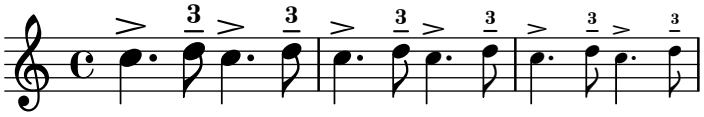
Questa sezione spiega come aggiungere enfasi agli elementi interni al rigo.

Scelta della dimensione del tipo di carattere

È possibile modificare la dimensione del tipo di carattere degli elementi della notazione. Tale modifica non altera la dimensione di altri simboli variabili, come le travature o le legature di portamento.
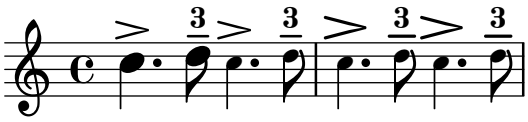
Nota: Per informazioni sulla dimensione del tipo di carattere per il testo, si veda ⟨undefined⟩ [Selecting font and font size], pagina ⟨undefined⟩.

```
\huge
c4.-> d8---3
\large
c4.-> d8---3
\normalsize
c4.-> d8---3
\small
c4.-> d8---3
\tiny
c4.-> d8---3
\teeny
c4.-> d8---3
```



Internamente, questi comandi impostano la proprietà `fontSize`. In questo modo la proprietà `font-size` viene impostata per tutti gli oggetti della formattazione. Il valore di `font-size` è un numero che indica la dimensione relativa alla dimensione standard per l'altezza del rigo corrente. Ogni grado in su corrisponde a un aumento di circa il 12% della dimensione del tipo di carattere. Sei gradi corrispondono esattamente a un fattore di due. La funzione Scheme `magstep` converte un numero di `font-size` in un fattore di ridimensionamento. Si può impostare la proprietà `font-size` anche direttamente, in modo da agire solo su certi oggetti di formattazione.

```
\set fontSize = #3
c4.-> d8---3
\override NoteHead.font-size = #-4
c4.-> d8---3
\override Script.font-size = #2
c4.-> d8---3
\override Stem.font-size = #-5
c4.-> d8---3
```



La modifica della dimensione del tipo di carattere si ottiene ridimensionando la dimensione, tra quelle predefinite, più vicina a quella desiderata. La dimensione standard (per `font-size = #0`) dipende dall'altezza standard del rigo: per un rigo di 20pt, viene scelto un tipo di carattere di 10pt.

La proprietà `font-size` si può impostare soltanto sugli oggetti di formattazione che usano i tipi di carattere, ovvero quegli oggetti che supportano l'interfaccia di formattazione `font-interface`.

Comandi predefiniti

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

Vedi anche
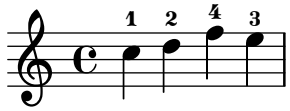
Frammenti: Sezione "Editorial annotations" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "font-interface" in *Guida al Funzionamento Interno*.

Indicazioni di diteggiatura
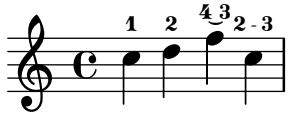
Le indicazioni di diteggiatura si inseriscono con '*nota-numero*':

```
c4-1 d-2 f-4 e-3
```



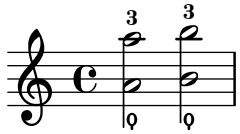
Si può usare il testo incluso dentro `\markup` o tra virgolette per indicare un cambio di dito.

```
c4-1 d-2 f\finger \markup \tied-lyric #"4~3" c\finger "2 - 3"
```
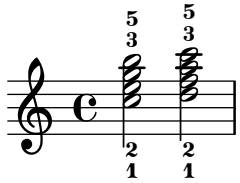


Si può aggiungere il simbolo del pollice per indicare che una nota deve essere suonata col pollice (ad esempio, nella musica per violoncello).

```
<a_\thumb a'-3>2 <b_\thumb b'-3>
```



È possibile indicare la diteggiatura di ogni singola nota di un accordo specificandola dopo ciascuna altezza.

```
<c-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
```



Le indicazioni di diteggiatura possono essere poste sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 585.

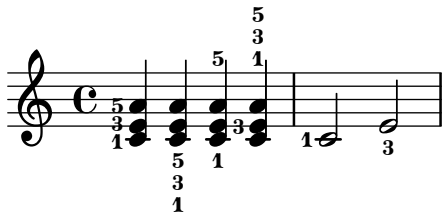
Frammenti di codice selezionati

Controllare il posizionamento delle diteggiature di un accordo

Il posizionamento dei numeri della diteggiatura può essere regolato in modo preciso. Perché l'orientamento funzioni, occorre usare il costrutto per gli accordi `<>` anche per una nota singola.

```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
```

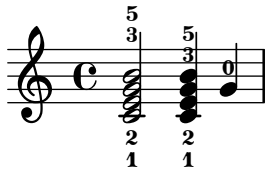
```
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Far sì che la diteggiatura appaia dentro il rigo

Per impostazione predefinita, le diteggiature orientate verticalmente sono poste fuori dal rigo. Tuttavia, questo comportamento può essere annullato. Attenzione: bisogna usare il costrutto per gli accordi `<>`, anche se si riferisce a una singola nota.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 <g'-0>
}
```



Evitare le collisioni con le diteggiature degli accordi

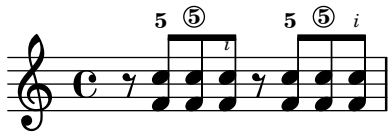
Diteggiature e numeri di corda applicati a note individuali evitano automaticamente le travature e i gambi, ma questo non vale per diteggiature e numeri di corda applicati alle singole note di un accordo. L'esempio seguente mostra come aggirare questo comportamento predefinito.

```
\relative c' {
  \set fingeringOrientations = #'(up)
  \set stringNumberOrientations = #'(up)
  \set strokeFingerOrientations = #'(up)

  % Default behavior
  r8
  <f c'-5>8
  <f c'\5>8
  <f c'-\rightHandFinger #2 >8

  % Corrected to avoid collisions
  r8
  \override Fingering.add-stem-support = ##t
  <f c'-5>8
  \override StringNumber.add-stem-support = ##t
  <f c'\5>8
  \override StrokeFinger.add-stem-support = ##t
  <f c'-\rightHandFinger #2 >8
```

```
}
```



Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 585.

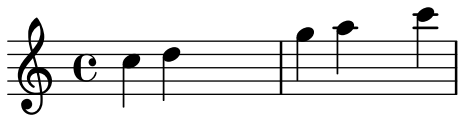
Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “FingeringEvent” in *Guida al Funzionamento Interno*, Sezione “fingering-event” in *Guida al Funzionamento Interno*, Sezione “Fingering_engraver” in *Guida al Funzionamento Interno*, Sezione “New_fingering_engraver” in *Guida al Funzionamento Interno*, Sezione “Fingering” in *Guida al Funzionamento Interno*.

Note nascoste

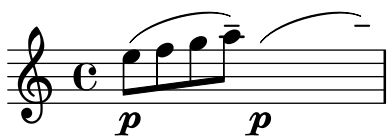
Le note nascoste (o invisibili o trasparenti) possono essere utili nella preparazione di esercizi di teoria e composizione.

```
c4 d
\hideNotes
e4 f
\unHideNotes
g a
\hideNotes
b
\unHideNotes
c
```



Questo comando rende invisibili le teste, i gambi e le code delle note, e le pause. Le travature sono invisibili se iniziano su una nota nascosta. Mentre gli oggetti attaccati a note invisibili sono comunque visibili.

```
e8(\p f g a)--
\hideNotes
e8(\p f g a)--
```



Comandi predefiniti

`\hideNotes`, `\unHideNotes`.

Vedi anche

Manuale d'apprendimento: Sezione "Visibilità e colore degli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Invisible rests], pagina ⟨undefined⟩, Sezione 5.4.6 [Visibility of objects], pagina 592, ⟨undefined⟩ [Hiding staves], pagina ⟨undefined⟩.

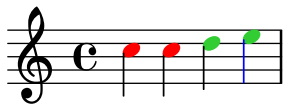
Frammenti: Sezione "Editorial annotations" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Note_spacing_engraver" in *Guida al Funzionamento Interno*, Sezione "NoteSpacing" in *Guida al Funzionamento Interno*.

Colorare gli oggetti

Si possono assegnare dei colori a ciascun oggetto. I nomi dei colori validi sono elencati nella Sezione A.7 [List of colors], pagina 630.

```
\override NoteHead.color = #red
c4 c
\override NoteHead.color = #(x11-color 'LimeGreen)
d
\override Stem.color = #blue
e
```
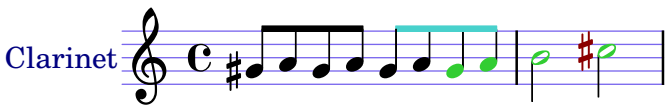


Si può accedere all'intera gamma di colori definita per X11 con la funzione Scheme `x11-color`. La funzione prende un argomento, che può essere un simbolo nella forma `'FooBar` o una stringa nella forma `"FooBar"`. La prima forma è più veloce da scrivere e più efficiente. Tuttavia, la seconda forma permette di accedere ai colori X11 attraverso la forma del nome che ha più di una parola.

La funzione `x11-color`, se non riesce a comprendere il parametro, restituisce il colore nero.

```
\override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}
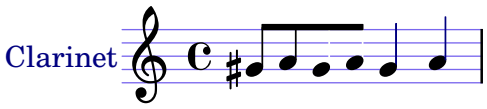
gis8 a
\override Beam.color = #(x11-color "medium turquoise")
gis a
\override Accidental.color = #(x11-color 'DarkRed)
gis a
\override NoteHead.color = #(x11-color "LimeGreen")
gis a
% questo parametro è volutamente assurdo; notare che i gambi restano neri
\override Stem.color = #(x11-color 'Boggle)
b2 cis
```



I colori RGB esatti si specificano con la funzione Scheme `rgb-color`.

```
\override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}

\override Stem.color = #(rgb-color 0 0 0)
gis8 a
\override Stem.color = #(rgb-color 1 1 1)
gis8 a
\override Stem.color = #(rgb-color 0 0 0.5)
gis4 a
```



Vedi anche

Guida alla notazione: Sezione A.7 [List of colors], pagina 630, Sezione 5.3.4 [The tweak command], pagina 579.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Problemi noti e avvertimenti

Un colore X11 non ha necessariamente la stessa identica tonalità di un normale colore dal nome simile.
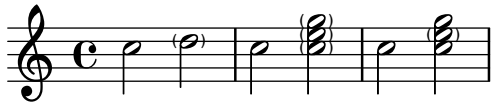
Non tutti i colori X11 sono distinguibili in un browser web. Per esempio, un browser potrebbe non mostrare alcuna differenza tra `LimeGreen` e `ForestGreen`. Per il web si consiglia di usare i colori normali (ovvero `blue`, `green`, `red`).

Le note in un accordo non possono essere colorate con un `\override`; al suo posto si usa `\tweak`, vedi Sezione 5.3.4 [The tweak command], pagina 579.

Parentesi
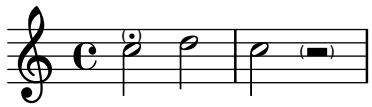
Gli oggetti possono essere messi tra parentesi se si usa il comando `\parenthesize` prima dell’evento musicale. Se precede un accordo, viene messa tra parentesi ogni nota dell’accordo. Si possono mettere tra parentesi anche singole note di un accordo.

```
c2 \parenthesize d
c2 \parenthesize <c e g>
c2 <c \parenthesize e g>
```



Si possono mettere tra parentesi anche oggetti diversi dalle note. Per le articolazioni è necessario usare un trattino prima del comando `\parenthesize`.

```
c2-\parenthesize -. d
c2 \parenthesize r
```



Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Parenthesis_engraver” in *Guida al Funzionamento Interno*, Sezione “ParenthesesItem” in *Guida al Funzionamento Interno*, Sezione “parentheses-interface” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Se si mette tra parentesi un accordo, viene creata una parentesi per ogni nota dell’accordo invece di una sola grande parentesi per l’intero accordo.

Gambi

Per ogni nota viene creato automaticamente un oggetto `Stem` (gambo). Vale anche per le semibrevi e le pause, anche se i loro gambi sono resi invisibili.

I gambi si possono posizionare sopra o sotto, vedi Sezione 5.4.2 [Direction and placement], pagina 585.

Comandi predefiniti
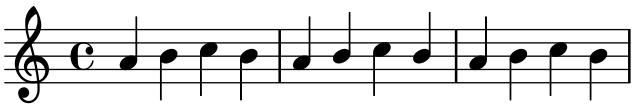
`\stemUp`, `\stemDown`, `\stemNeutral`.

Frammenti di codice selezionati

Direzione predefinita dei gambi sulla linea centrale del rigo

La direzione predefinita dei gambi sulla linea centrale del rigo si imposta con la proprietà `neutral-direction` dell’oggetto `Stem`.

```
\relative c'' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}
```
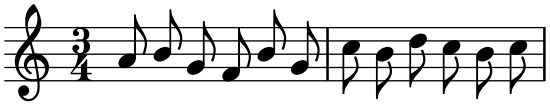


Cambiare automaticamente la direzione del gambo della nota centrale in base alla melodia

LilyPond può modificare la direzione del gambo della nota centrale di un rigo in modo che segua la melodia: occorre aggiungere l’incisore `Melody_engraver` al contesto Voice e sovrascrivere la proprietà `neutral-direction` di Stem.

```
\relative c'' {
  \time 3/4
  \autoBeamOff
  a8 b g f b g |
  c b d c b c
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
```

```
    \override Stem.neutral-direction = #'()
  }
}
```



Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 585.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Stem_engraver” in *Guida al Funzionamento Interno*, Sezione “Stem” in *Guida al Funzionamento Interno*, Sezione “stem-interface” in *Guida al Funzionamento Interno*.
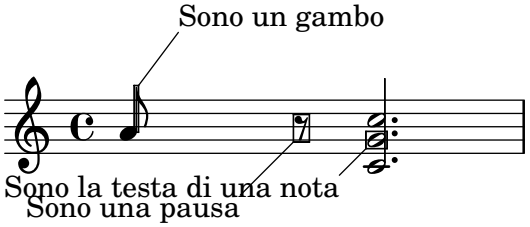
1.7.2 Esterne al rigo

Questa sezione spiega come dare risalto agli elementi nel rigo attraverso delle note esterne al rigo.

Nuvoletta di aiuto

Si possono contrassegnare e nominare gli elementi della notazione tramite una nuvoletta quadrata. La sua funzione principale è spiegare la notazione.

```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Sono un gambo" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Sono una pausa" }
  r
  <c, g'-\balloonText #'(-2 . -2) \markup { "Sono la testa di una nota" } c>2.
}
```

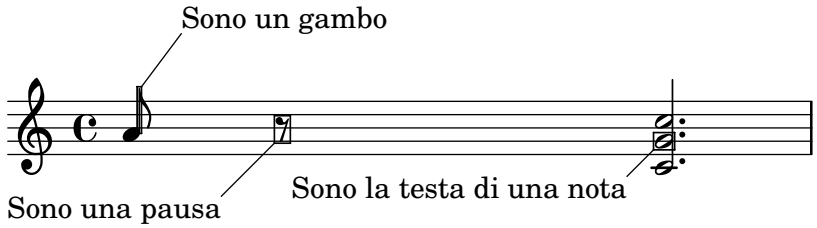


Ci sono due funzioni musicali, `balloonGrobText` e `balloonText`; la prima si usa nella forma `\once \override` per attaccare del testo a un qualsiasi oggetto grafico (grob), mentre la seconda viene usata come `\tweak`, solitamente all’interno degli accordi, per attaccare del testo a una singola nota.

Il testo nella nuvoletta influenza la spaziatura delle note, ma è possibile modificare questo comportamento:

```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Sono un gambo" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Sono una pausa" }
  r
```

```
  \balloonLengthOn
  <c, g'-\balloonText #'(-2 . -2) \markup { "Sono la testa di una nota" } c>2.
}
```



Comandi predefiniti

`\balloonLengthOn`, `\balloonLengthOff`.

Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Balloon_engraver” in *Guida al Funzionamento Interno*, Sezione “BalloonTextItem” in *Guida al Funzionamento Interno*, Sezione “balloon-interface” in *Guida al Funzionamento Interno*.

Linee della griglia

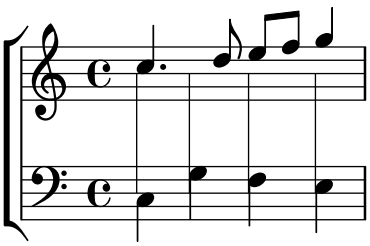
Si possono disegnare delle linee verticali tra i righi sincronizzate con le note.

Si deve usare l’incisore `Grid_point_engraver` per creare le estremità delle linee, mentre l’incisore `Grid_line_span_engraver` serve a disegnare le linee. Per impostazione predefinita, le linee della griglia sono allineate orizzontalmente sotto e sul lato sinistro delle teste di nota. Le linee si estendono a partire dalle linee centrali di ciascun rigo. `gridInterval` deve specificare la durata che separa le linee.

```
\layout {
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #(ly:make-moment 1/4)
  }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
  }
}

\score {
  \new ChoirStaff <<
    \new Staff \relative c'' {
      \stemUp
      c4. d8 e8 f g4
    }
    \new Staff \relative c {
      \clef bass
      \stemDown
      c4 g' f e
    }
```
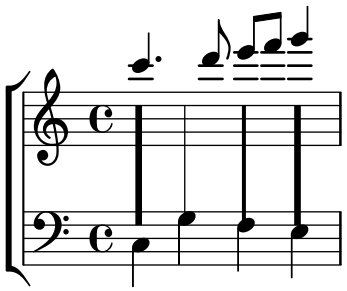
```
  >>
}
```



Frammenti di codice selezionati

Modificare l'aspetto delle linee della griglia

L'aspetto delle linee della griglia può essere modificato sovrascrivendo alcune delle loro proprietà.

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \relative c'' {
        \stemUp
        c'4. d8 e8 f g4
      }
    }
    \new Staff {
      \relative c {
        % this moves them up one staff space from the default position
        \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
        \stemDown
        \clef bass
        \once \override Score.GridLine.thickness = #5.0
        c4
        \once \override Score.GridLine.thickness = #1.0
        g'4
        \once \override Score.GridLine.thickness = #3.0
        f4
        \once \override Score.GridLine.thickness = #5.0
        e4
      }
    }
  >>
  \layout {
    \context {
      \Staff
      % set up grids
      \consists "Grid_point_engraver"
      % set the grid interval to one quarter note
      gridInterval = #(ly:make-moment 1/4)
    }
    \context {
      \Score
      \consists "Grid_line_span_engraver"
```

```
      % this moves them to the right half a staff space
      \override NoteColumn.X-offset = #-0.5
    }
  }
}
```



Vedi anche
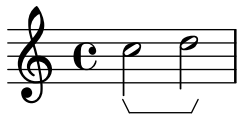
Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Grid_line_span_engraver” in *Guida al Funzionamento Interno*, Sezione “Grid_point_engraver” in *Guida al Funzionamento Interno*, Sezione “GridLine” in *Guida al Funzionamento Interno*, Sezione “GridPoint” in *Guida al Funzionamento Interno*, Sezione “grid-line-interface” in *Guida al Funzionamento Interno*, Sezione “grid-point-interface” in *Guida al Funzionamento Interno*.

Parentesi analitiche

Nell’analisi musicale si usano le parentesi per indicare la struttura dei brani musicali. Sono supportate delle semplici parentesi orizzontali.

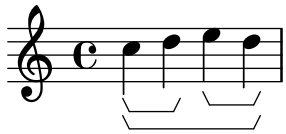
```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' {
  c2\startGroup
  d\stopGroup
}
```



Le parentesi analitiche si possono annidare.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' {
  c4\startGroup\startGroup
```

```
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```
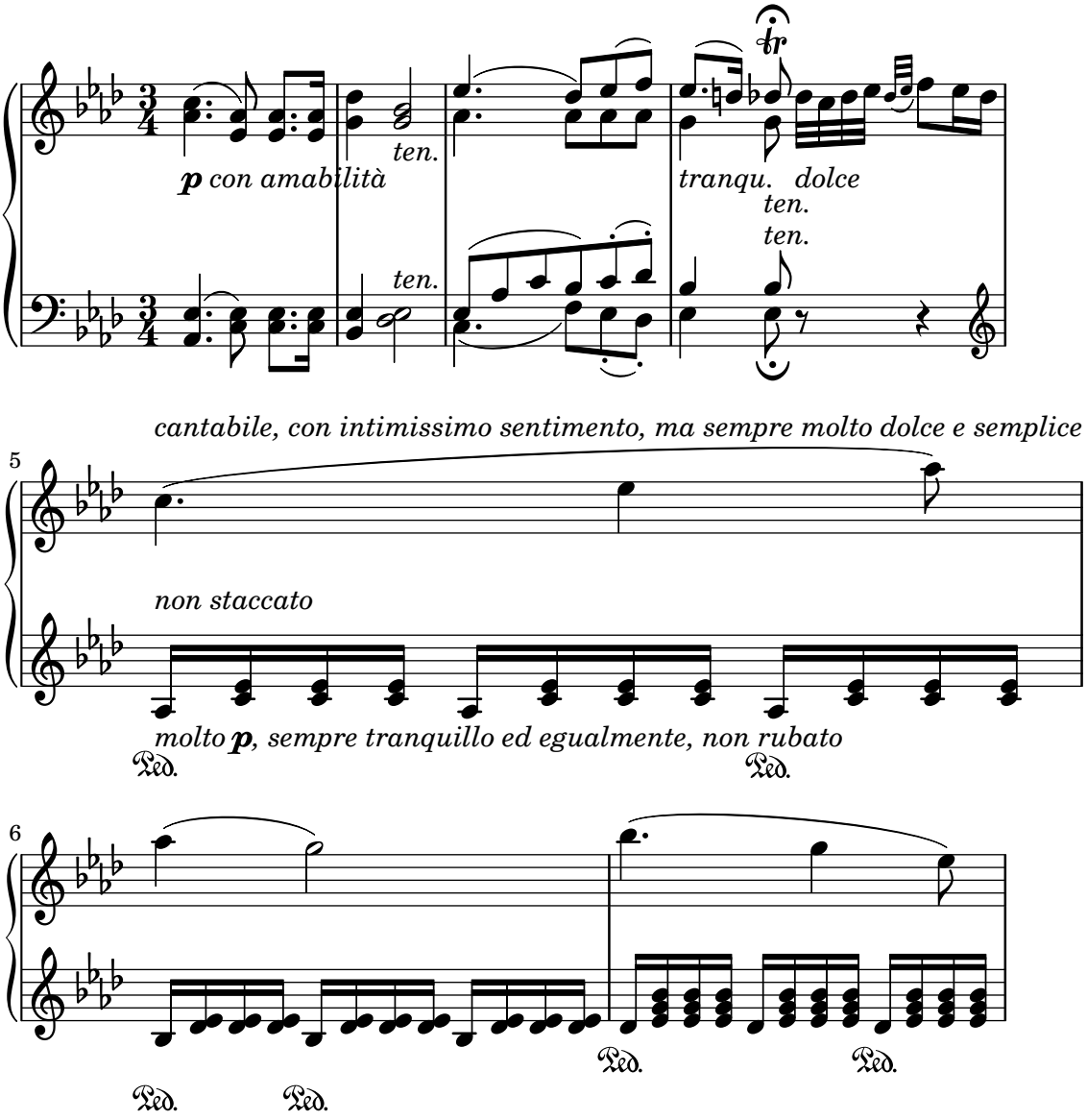


Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Horizontal_bracket_engraver” in *Guida al Funzionamento Interno*, Sezione “HorizontalBracket” in *Guida al Funzionamento Interno*, Sezione “horizontal-bracket-interface” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

1.8 Testo



Questa sezione spiega come includere del testo (con vari tipi di formattazione) nelle partiture musicali.

Alcuni elementi testuali che non sono trattati qui sono discussi in altre sezioni specifiche: Sezione 2.1 [Vocal music], pagina 247, Sezione 3.2 [Titles and headers], pagina 456.

1.8.1 Inserimento del testo
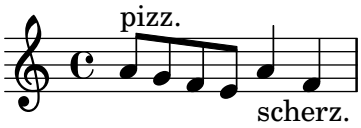
Questa sezione presenta vari modi di aggiungere del testo a una partitura.

Nota: Per scrivere caratteri accentati e speciali (come quelli di altre lingue), basta inserire semplicemente i caratteri nel file LilyPond, purché il file sia salvato in formato UTF-8. Ulteriori informazioni in [Text encoding], pagina 486.

Scritte
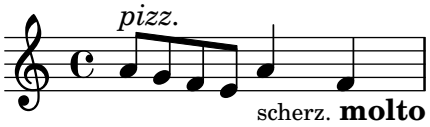
Si possono aggiungere a una partitura delle semplici indicazioni con del "testo tra virgolette", come mostrato nell'esempio seguente. Tali indicazioni possono essere posizionate sopra o sotto il rigo, usando la sintassi descritta in Sezione 5.4.2 [Direction and placement], pagina 585.

```
a8^"pizz." g f e a4-"scherz." f
```
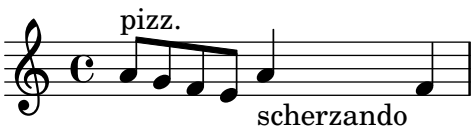


In realtà questa sintassi è una scorciatoia; si può specificare una formattazione del testo più complessa usando in modo esplicito un blocco `\markup`, come è spiegato in ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

```
a8^\markup { \italic pizz. } g f e
a4_\markup { \tiny scherz. \bold molto } f
```



Le indicazioni testuali, di norma, non influenzano la spaziatura delle note. Ma è possibile far sì che la loro larghezza venga presa in considerazione: nell'esempio seguente la prima stringa di testo non influenza la spaziatura, mentre la seconda sì.

```
a8^"pizz." g f e
\textLengthOn
a4_"scherzando" f
```



Oltre alle scritte, si possono attaccare alle note anche le articolazioni. Ulteriori informazioni in ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.

Per maggiori informazioni sull'ordinamento relativo delle scritte e delle articolazioni si veda Sezione "Posizionamento degli oggetti" in *Manuale di Apprendimento*.

Comandi predefiniti

`\textLengthOn`, `\textLengthOff`.

Vedi anche

Manuale d'apprendimento: Sezione "Posizionamento degli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, Sezione 5.4.2 [Direction and placement], pagina 585, ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

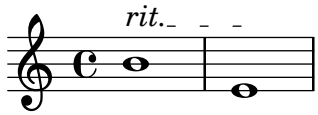
Per verificare che le scritte e il testo vocale siano entro i margini occorrono ulteriori calcoli. Nei casi in cui è richiesta un'esecuzione leggermente più veloce, usare

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Estensori del testo
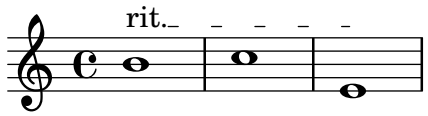
Alcune indicazioni esecutive, per esempio *rallentando* o *accelerando*, appaiono in forma testuale e vengono estese lungo molteplici note con delle linee punteggiate. Tali oggetti, chiamati "estensori" (spanner), si creano collegando due note con la seguente sintassi:

```
\override TextSpanner.bound-details.left.text = "rit."
b1\startTextSpan
e,\stopTextSpan
```



La stringa testuale da stampare viene impostata attraverso le proprietà dell'oggetto. Per impostazione predefinita, appare in corsivo, ma si può ottenere una formattazione diversa tramite i blocchi `\markup`, come è spiegato in ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

```
\override TextSpanner.bound-details.left.text =
  \markup { \upright "rit." }
b1\startTextSpan c
e,\stopTextSpan
```



Lo stile della linea, così come la stringa testuale, può essere definito come una proprietà dell'oggetto. Questa sintassi è descritta in Sezione 5.4.7 [Line styles], pagina 598.

Comandi predefiniti

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.

Problemi noti e avvertimenti

LilyPond è capace di gestire un solo estensore del testo per ogni voce.
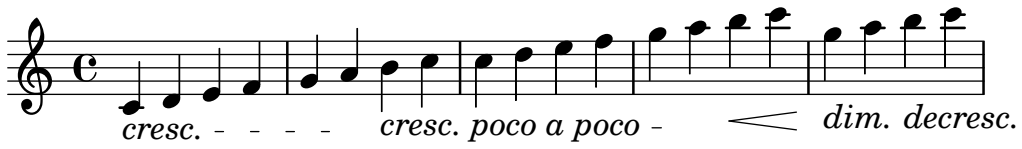
Frammenti di codice selezionati

Estensore testuale della dinamica personalizzato

Si possono definire estensori testuali personalizzati che fanno uso delle forcine e dei crescendo testuali. \< e \> generano le forcine, \cresc etc. generano gli estensori testuali.

```
% Some sample text dynamic spanners, to be used as postfix operators
crpoco =
#(make-music 'CrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "cresc. poco a poco")

\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decresc c4\!
}
```

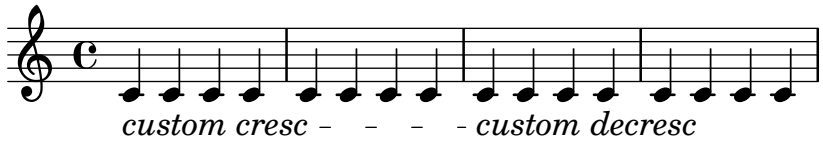


Estensore testuale della dinamica personalizzato

Funzioni postfix per estensori testuali personalizzati del crescendo. Gli estensori devono iniziare sulla prima nota della misura; e bisogna usare -\mycresc, altrimenti l'inizio dell'estensore viene assegnato alla nota successiva.

```
% Two functions for (de)crescendo spanners where you can explicitly give the
% spanner text.
mycresc =
#(define-music-function (parser location mymarkup) (markup?)
   (make-music 'CrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))
mydecresc =
#(define-music-function (parser location mymarkup) (markup?)
   (make-music 'DecrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4\! c4 c4
}
```



Vedi anche

Guida alla notazione: Sezione 5.4.7 [Line styles], pagina 598, ⟨undefined⟩ [Dynamics], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.
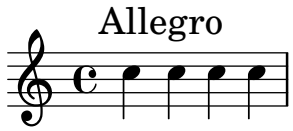
Frammenti: Sezione "Text" in *Frammenti di codice*, Sezione "Expressive marks" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextSpanner" in *Guida al Funzionamento Interno*.

Indicazioni testuali
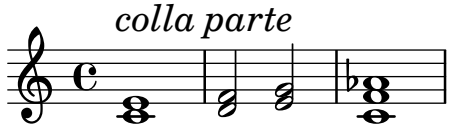
Si possono aggiungere vari elementi testuali a una partitura tramite la sintassi descritta in ⟨undefined⟩ [Rehearsal marks], pagina ⟨undefined⟩:

```
c4
\mark "Allegro"
c c c
```
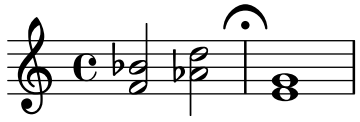


Questa sintassi permette di porre del testo sopra una stanghetta; una formattazione del testo più complessa è possibile grazie al blocco `\markup`, come è spiegato in ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩:

```
<c e>1
\mark \markup { \italic { colla parte } }
<d f>2 <e g>
<c f aes>1
```



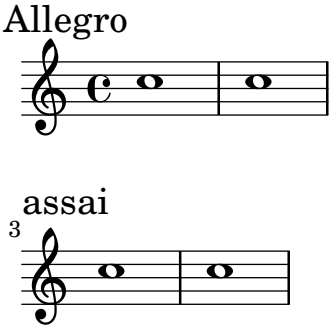
Questa sintassi permette anche di stampare segni speciali, come coda, segno o corona, se si specifica il nome appropriato del simbolo, come è spiegato in ⟨undefined⟩ [Music notation inside markup], pagina ⟨undefined⟩:

```
<bes f>2 <aes d>
\mark \markup { \musicglyph #"scripts.ufermata" }
<e g>1
```



Tali oggetti vengono posizionati soltanto sopra il rigo superiore della partitura; a seconda che siano inseriti alla fine o a metà di una battuta, possono trovarsi sopra la stanghetta o tra le note. Se inserito prima di un'interruzione di linea, l'indicazione apparirà all'inizio della linea successiva.

```
\mark "Allegro"
c1 c
\mark "assai" \break
c  c
```



Comandi predefiniti

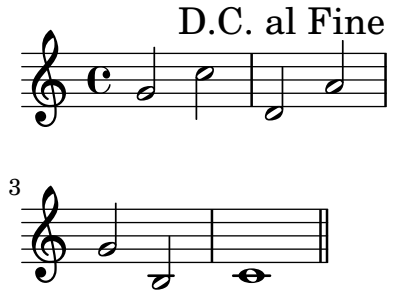
`\markLengthOn`, `\markLengthOff`.

Frammenti di codice selezionati

Posizionare le indicazioni alla fine di una linea

È possibile posizionare le indicazioni alla fine della linea corrente, invece che all'inizio della linea successiva. In tali casi, può essere preferibile allineare l'estremità destra dell'indicazione alla stanghetta.

```
\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark.break-visibility = #end-of-line-visible
  \once \override Score.RehearsalMark.self-alignment-X = #RIGHT
  \mark "D.C. al Fine"
  \break
  g2 b,
  c1 \bar "||"
}
```

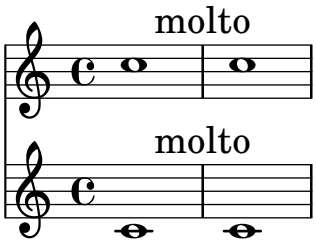


Stampare le indicazioni su ogni rigo

Sebbene le indicazioni testuali siano di norma collocate solo sopra il rigo più alto, è possibile farle apparire su ogni rigo.

```
\score {
  <<
    \new Staff { c''1 \mark "molto" c'' }
    \new Staff { c'1 \mark "molto" c' }
```

```
  >>
  \layout {
    \context {
      \Score
      \remove "Mark_engraver"
      \remove "Staff_collecting_engraver"
    }
    \context {
      \Staff
      \consists "Mark_engraver"
      \consists "Staff_collecting_engraver"
    }
  }
}
```



Vedi anche

Guida alla notazione: ⟨undefined⟩ [Rehearsal marks], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, ⟨undefined⟩ [Music notation inside markup], pagina ⟨undefined⟩, Sezione A.8 [The Feta font], pagina 632.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MarkEvent” in *Guida al Funzionamento Interno*, Sezione “Mark_engraver” in *Guida al Funzionamento Interno*, Sezione “RehearsalMark” in *Guida al Funzionamento Interno*.

Testo separato

Un blocco `\markup` può esistere di per sé, fuori da qualsiasi blocco `\score`, come un’ “espressione di livello superiore”. Questa sintassi è descritta in Sezione 3.1.5 [File structure], pagina 454.

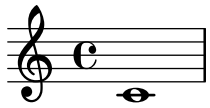
```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

Tomorrow, and tomorrow, and tomorrow...

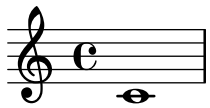
Ciò permette di stampare il testo in modo autonomo dalla musica, ed è utile soprattutto quando il file di input contiene vari brani musicali, come è spiegato in Sezione 3.1.2 [Multiple scores in a book], pagina 451.

```
\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

```
\score {
  c'1
}
```



Tomorrow, and tomorrow, and tomorrow...



Blocchi di testo separati possono essere estesi per molte pagine, rendendo possibile la realizzazione di documenti o libri interamente con LilyPond. Questa funzionalità, e la sintassi specifica che richiede, è descritta in ⟨undefined⟩ [Multi-page markup], pagina ⟨undefined⟩.

Comandi predefiniti

`\markup`, `\markuplist`.

Frammenti di codice selezionati

Testo separato su due colonne

Il testo separato può essere disposto su varie colonne con i comandi di `\markup`:

```
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { O sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column \italic {
      \line { O sacred feast }
      \line { in which Christ is received, }
      \line { the memory of His Passion is renewed, }
      \line { the mind is filled with grace, }
      \line { and a pledge of future glory is given to us. }
      \line { Amen. }
    }
    \hspace #1
  }
}
```

O sacrum convivium	*O sacred feast*
in quo Christus sumitur,	*in which Christ is received,*
recolitur memoria passionis ejus,	*the memory of His Passion is renewed,*
mens impletur gratia,	*the mind is filled with grace,*
futurae gloriae nobis pignus datur.	*and a pledge of future glory is given to us.*
Amen.	*Amen.*

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, Sezione 3.1.5 [File structure], pagina 454, Sezione 3.1.2 [Multiple scores in a book], pagina 451, ⟨undefined⟩ [Multi-page markup], pagina ⟨undefined⟩.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

1.8.2 Formattazione del testo

Questa sezione presenta la formattazione del testo basilare e quella avanzata, usando la sintassi specifica della modalità `\markup`.

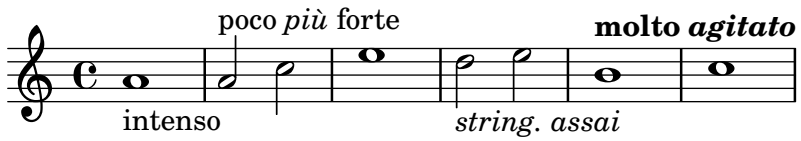
Introduzione al testo a margine

Un blocco `\markup` permette di comporre del testo con un'ampia sintassi chiamata "modalità markup".

La sintassi di markup è simile alla solita sintassi di LilyPond: un'espressione `\markup` viene racchiusa tra parentesi graffe `{... }`. Una singola parola viene considerata un'espressione minima, e quindi non è necessario racchiuderla tra parentesi.
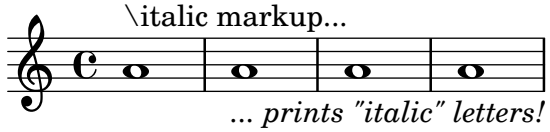
Diversamente dalle indicazioni testuali "tra virgolette", i blocchi `\markup` possono contenere espressioni o comandi di markup annidati, inseriti col carattere di barra inversa `\`. Tali comandi hanno effetto solo sulla prima espressione che segue.

```
a1-\markup intenso
a2^\markup { poco \italic più forte  }
c e1
d2_\markup { \italic "string. assai" }
e
b1^\markup { \bold { molto \italic  agitato } }
c
```



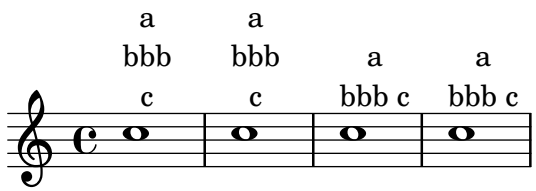
Un blocco `\markup` può contenere anche stringhe di testo tra virgolette. Tali stringhe vengono trattate come espressioni testuali minime, e quindi qualsiasi comando di markup o carattere speciale (come `\` e `#`) apparirà alla lettera senza influenzare la formattazione del testo. Le stesse doppie virgolette possono essere stampate facendole precedere da una barra inversa.

```
a1^"\italic markup..."
a_\markup { \italic "... prints \"italic\" letters!" }
a a
```



Perché sia trattata come un'espressione distinta, una lista di parole deve essere racchiusa tra virgolette doppie o preceduta da un comando. Il modo in cui le espressioni musicali sono definite influenza il modo in cui saranno sistemate, centrate e allineate; nell'esempio seguente, la seconda espressione di `\markup` viene trattata nello stesso modo della prima:
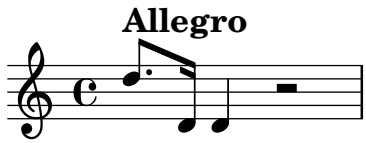
```
c1^\markup { \center-column { a bbb c } }
c1^\markup { \center-column { a { bbb c } } }
c1^\markup { \center-column { a \line { bbb c } } }
c1^\markup { \center-column { a "bbb c" } }
```



I markup possono essere salvati in delle variabili, che possono poi essere attaccate direttamente alle note:

```
allegro = \markup { \bold \large Allegro }

{
  d''8.^\allegro
  d'16 d'4 r2
}
```



Una lista completa dei comandi specifici di `\markup` si trova in Sezione A.10 [Text markup commands], pagina 653.

Vedi anche

Guida alla notazione: Sezione A.10 [Text markup commands], pagina 653.

Frammenti: Sezione "Text" in *Frammenti di codice*.

File installati: '`scm/markup.scm`'.
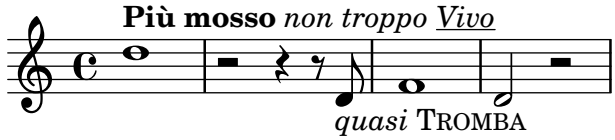
Problemi noti e avvertimenti

Gli errori di sintassi relativi alla modalità markup possono essere poco chiari.

Scelta del tipo di carattere e della dimensione

La modalità markup permette di cambiare il tipo di carattere:

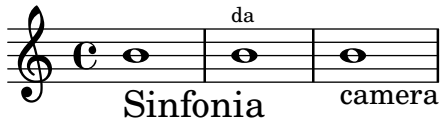
```
d1^\markup {
  \bold { Più mosso }
  \italic { non troppo \underline Vivo }
}
r2 r4 r8
d,_\markup { \italic quasi \smallCaps Tromba }
f1 d2 r
```



Si può modificare la dimensione del tipo di carattere, rispetto alla dimensione globale del rigo, in vari modi.

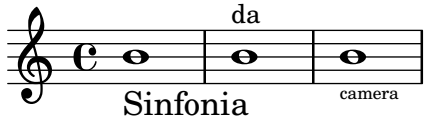
Si può impostare su una dimensione predefinita,

```
b1_\markup { \huge Sinfonia }
b1^\markup { \teeny da }
b1-\markup { \normalsize camera }
```



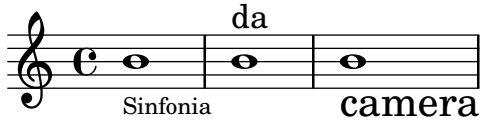
oppure in modo proporzionale rispetto al valore precedente,

```
b1_\markup { \larger Sinfonia }
b1^\markup { \smaller da }
b1-\markup { \magnify #0.6  camera }
```



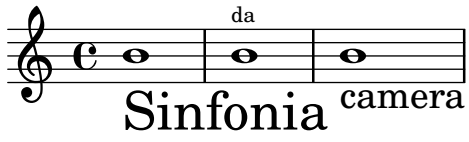
Può essere aumentata o diminuita rispetto al valore impostato per la dimensione globale del rigo:

```
b1_\markup { \fontsize #-2 Sinfonia }
b1^\markup { \fontsize #1 da }
b1-\markup { \fontsize #3 camera }
```



Si può impostare anche su una dimensione fissa (in punti), indipendentemente dalla dimensione globale del rigo:

```
b1_\markup { \abs-fontsize #20 Sinfonia }
b1^\markup { \abs-fontsize #8 da }
b1-\markup { \abs-fontsize #14 camera }
```



È possibile stampare il testo come pedice o apice. Per impostazione predefinita, questo appaiono in corpo più piccolo, ma si può usare anche un corpo normale:

```
\markup {
  \column {
    \line { 1 \super st movement }
    \line { 1 \normal-size-super st movement
      \sub { (part two) }  }
```

```
  }
}
```

1st movement
1st movement (part two)

La modalità di markup fornisce un modo semplice per scegliere famiglie di caratteri diverse. Se non specificato altrimenti, viene scelto automaticamente il carattere tipografico con grazie (il tipo romano); nell'ultima linea dell'esempio seguente non c'è differenza tra la prima e la seconda parola.

```
\markup {
  \column {
    \line { Act \number 1 }
    \line { \sans { Scene I. } }
    \line { \typewriter { Verona. An open place. } }
    \line { Enter \roman Valentine and Proteus. }
  }
}
```

Act 1
Scene I.
Verona. An open place.
Enter Valentine and Proteus.

Alcune di queste famiglie di caratteri, usate per elementi specifici come i numeri o le dinamiche, non forniscono tutti i caratteri, come accennato in ⟨undefined⟩ [New dynamic marks], pagina ⟨undefined⟩ e ⟨undefined⟩ [Manual repeat marks], pagina ⟨undefined⟩.

Se usati all'interno di una parola, alcuni comandi che cambiano il tipo di carattere o la formattazione potrebbero produrre uno spazio vuoto indesiderato. Si può facilmente risolvere concatenando insieme gli elementi testuali:

```
\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}
```

1st movement
p, *con dolce espressione*

Una lista completa dei comandi per cambiare il tipo di carattere o per usare tipi di carattere personalizzati si trova in Sezione A.10.1 [Font], pagina 653.

È possibile anche definire i propri gruppi di tipi di carattere, come è spiegato in ⟨undefined⟩ [Fonts], pagina ⟨undefined⟩.

Comandi predefiniti

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

Vedi anche

Guida alla notazione: Sezione A.10.1 [Font], pagina 653, ⟨undefined⟩ [New dynamic marks], pagina ⟨undefined⟩, ⟨undefined⟩ [Manual repeat marks], pagina ⟨undefined⟩, ⟨undefined⟩ [Fonts], pagina ⟨undefined⟩.

File installati: '`scm/define-markup-commands.scm`'.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

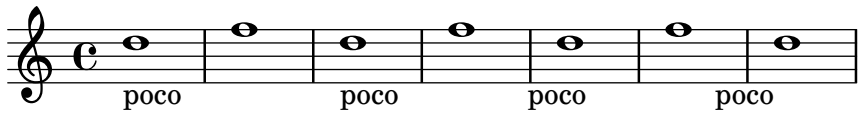
L'uso dei comandi di dimensionamento dei caratteri `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large` e `\huge` produce una spaziatura della linea imprevedibile rispetto all'uso di `\fontsize`.

Allineamento del testo

Questa sottosezione spiega come posizionare il testo nella modalità markup. Gli oggetti markup possono anche essere spostati interamente tramite la sintassi descritta in Sezione "Spostare gli oggetti" in *Manuale di Apprendimento*.

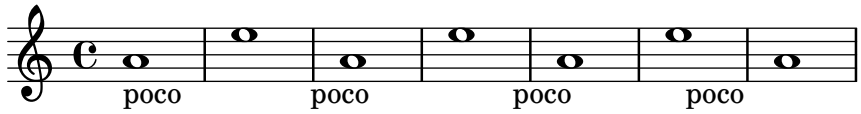
Gli oggetti di markup possono essere allineati in vari modi. Per impostazione predefinita, l'indicazione testuale è allineata rispetto al suo margine sinistro: nell'esempio seguente, non c'è differenza tra il primo e il secondo markup.

```
d1-\markup { poco }
f
d-\markup { \left-align poco }
f
d-\markup { \center-align { poco } }
f
d-\markup { \right-align poco }
```



L'allineamento orizzontale può essere ritoccato usando un valore numerico:

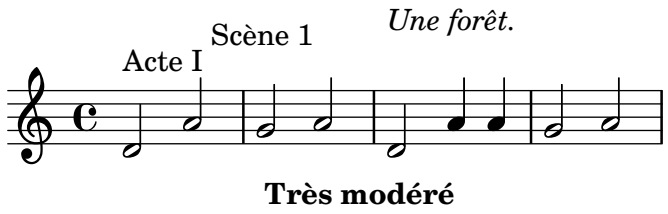
```
a1-\markup { \halign #-1 poco }
e'
a,-\markup { \halign #0 poco }
e'
a,-\markup { \halign #0.5 poco }
e'
a,-\markup { \halign #2 poco }
```



Alcuni oggetti possono avere proprie procedure di allineamento, e dunque non sono influenzate da questi comandi. È possibile spostare tali oggetti di markup tutti insieme, come mostrato ad esempio in ⟨undefined⟩ [Text marks], pagina ⟨undefined⟩.

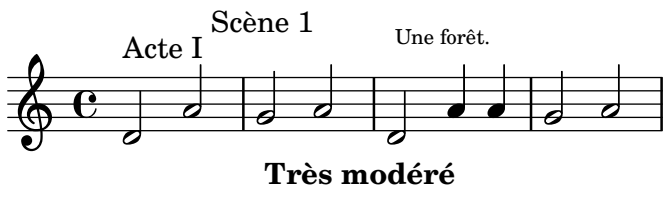
L'allineamento verticale è un po' più complesso. Come si è detto prima, gli oggetti di markup possono essere spostati tutti insieme; tuttavia è anche possibile spostare elementi specifici all'interno di un blocco markup. In questo caso l'elemento da spostare deve essere preceduto da un *punto di riferimento*, che può essere un altro elemento markup o un oggetto invisibile. L'esempio seguente illustra queste due possibilità; l'ultimo markup in questo esempio non ha un punto di riferimento e di conseguenza non si muove.

```
d2^\markup {
  Acte I
  \raise #2 { Scène 1 }
}
a'
g_\markup {
  \null
  \lower #4 \bold { Très modéré }
}
a
d,^\markup {
  \raise #4 \italic { Une forêt. }
}
a'4 a g2 a
```



Alcuni comandi possono cambiare l'allineamento sia orizzontale che verticale degli oggetti testuali in modalità markup. Qualsiasi oggetto interessato da questi comandi deve essere preceduto da un punto di riferimento:

```
d2^\markup {
  Acte I
  \translate #'(-1 . 2) "Scène 1"
}
a'
g_\markup {
  \null
  \general-align #Y #3.2 \bold "Très modéré"
}
a
d,^\markup {
  \null
  \translate-scaled #'(-1 . 2) \teeny "Une forêt."
}
a'4 a g2 a
```



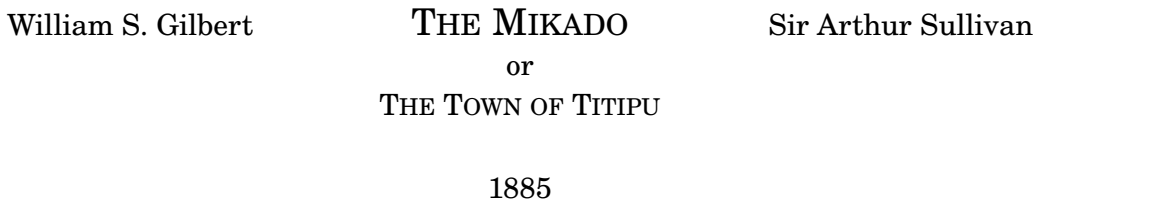
Un oggetto markup può includere varie linee di testo. Nell'esempio seguente, ogni elemento o espressione viene posizionato sulla sua linea, allineato a sinistra o centrato:

```
\markup {
  \column {
    a
    "b c"
    \line { d e f }
  }
  \hspace #10
  \center-column {
    a
    "b c"
    \line { d e f }
  }
}
```



Analogalmente, una lista di elementi o espressioni può essere distesa per riempire l'intera larghezza orizzontale della linea (se c'è un solo elemento, verrà centrato sulla pagina). Queste espressioni possono a loro volta includere del testo multilinea o una qualsiasi altra espressione di markup:

```
\markup {
  \fill-line {
    \line { William S. Gilbert }
    \center-column {
      \huge \smallCaps "The Mikado"
      or
      \smallCaps "The Town of Titipu"
    }
    \line { Sir Arthur Sullivan }
  }
}
\markup {
  \fill-line { 1885 }
}
```



Indicazioni testuali lunghe possono andare a capo automaticamente in base alla larghezza della linea specificata. Possono essere allineate a sinistra o giustificate, come mostra l'esempio seguente.

```
\markup {
  \column {
    \line  \smallCaps { La vida breve }
```

```
    \line \bold { Acto I }
    \wordwrap \italic {
      (La escena representa el corral de una casa de
      gitanos en el Albaicín de Granada.  Al fondo una
      puerta por la que se ve el negro interior de
      una Fragua, iluminado por los rojos resplandores
      del fuego.)
    }
    \hspace #0

    \line \bold { Acto II }
    \override #'(line-width . 50)
    \justify \italic {
      (Calle de Granada.  Fachada de la casa de Carmela
      y su hermano Manuel con grandes ventanas abiertas
      a través de las que se ve el patio
      donde se celebra una alegre fiesta)
    }
  }
}
```

La vida breve

Acto I

(La escena representa el corral de una casa de gitanos en el Albaicín de Granada. Al fondo una puerta por la que se ve el negro interior de una Fragua, iluminado por los rojos resplandores del fuego.)

Acto II

(Calle de Granada. Fachada de la casa de Carmela y su hermano Manuel con grandes ventanas abiertas a través de las que se ve el patio donde se celebra una alegre fiesta)

Una lista completa dei comandi di allineamento del testo si trova in Sezione A.10.2 [Align], pagina 662.

Vedi anche

Manuale d'apprendimento: Sezione "Spostare gli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione A.10.2 [Align], pagina 662, ⟨undefined⟩ [Text marks], pagina ⟨undefined⟩.

File installati: '`scm/define-markup-commands.scm`'.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

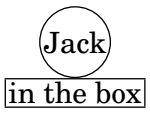
Notazione grafica nel blocco markup

Si possono aggiungere vari oggetti grafici a una partitura attraverso i comandi di markup.

Alcuni comandi di markup consentono di decorare gli elementi testuali con degli elementi grafici, come è illustrato nell'esempio seguente.

```
\markup \fill-line {
  \center-column {
```

```
    \circle Jack
    \box "in the box"
    \null
    \line {
      Erik Satie
      \hspace #3
      \bracket "1866 - 1925"
    }
    \null
    \rounded-box \bold Prelude
  }
}
```



Erik Satie [1866 - 1925]

Prelude

Alcuni comandi possono richiedere un aumento del padding intorno al testo; per farlo si usano dei comandi di markup, descritti in modo esaustivo in Sezione A.10.2 [Align], pagina 662.

```
\markup \fill-line {
  \center-column {
    \box "Charles Ives (1874 - 1954)"
    \null
    \box \pad-markup #2 "THE UNANSWERED QUESTION"
    \box \pad-x #8 "A Cosmic Landscape"
    \null
  }
}
\markup \column {
  \line {
    \hspace #10
    \box \pad-to-box #'(-5 . 20) #'(0 . 5)
      \bold "Largo to Presto"
  }
  \pad-around #3
      "String quartet keeps very even time,
Flute quartet keeps very uneven time."
}
```

Charles Ives (1874 - 1954)

THE UNANSWERED QUESTION

A Cosmic Landscape

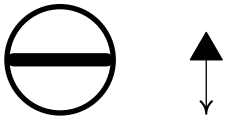
Largo to Presto

String quartet keeps very even time, Flute quartet keeps very uneven time.

Si possono produrre altri elementi grafici o simboli che non richiedono alcun testo. Come con qualsiasi espressione di markup, tali oggetti possono essere combinati.

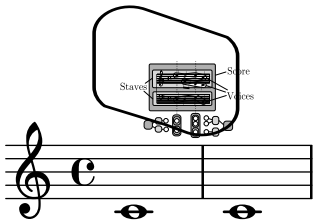
```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Le funzionalità grafiche avanzate comprendono la possibilità di includere file di immagini convertite nel formato Encapsulated PostScript (*eps*), oppure di inserire la grafica direttamente nel file di input, usando del codice PostScript nativo. In tal caso, può essere utile specificare esplicitamente la dimensione del disegno, come è mostrato sotto:

```
c1^\markup {
  \combine
    \epsfile #X #10 #"./context-example.eps"
    \with-dimensions #'(0 . 6) #'(0 . 10)
    \postscript #"
      -2 3 translate
      2.7 2 scale
      newpath
      2 -1 moveto
      4 -2 4 1 1 arct
      4 2 3 3 1 arct
      0 4 0 3 1 arct
      0 0 1 -1 1 arct
      closepath
      stroke"
  }
c
```



Una lista completa dei comandi specifici per la grafica si trova in Sezione A.10.3 [Graphic], pagina 677.

Vedi anche

Guida alla notazione: Sezione A.10.3 [Graphic], pagina 677, ⟨undefined⟩ [Editorial annotations], pagina ⟨undefined⟩, Sezione A.10.2 [Align], pagina 662.

File installati: ‘`scm/define-markup-commands.scm`’, ‘`scm/stencil.scm`’.

Frammenti: Sezione “Text” in *Frammenti di codice*.

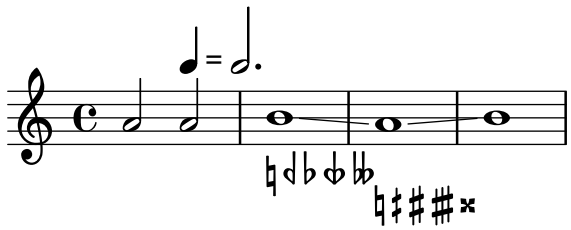
Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

Notazione musicale nel blocco markup

Si possono aggiungere vari elementi della notazione musicale dentro un oggetto markup.

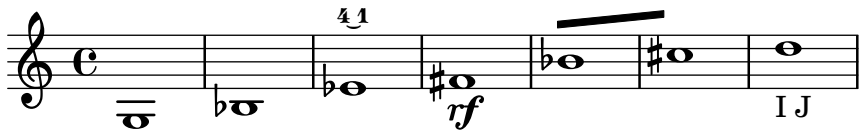
Per le note e le alterazioni esistono dei comandi markup appositi:

```
a2 a^\markup {
  \note #"4" #1
  =
  \note-by-number #1 #1 #1.5
}
b1_\markup {
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a1_\markup {
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b
```



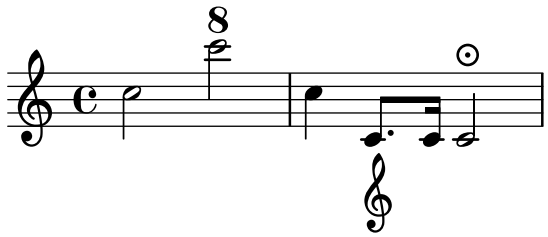
Anche altri oggetti della notazione possono essere stampati in modalità markup:

```
g1 bes
ees\finger \markup \tied-lyric #"4~1"
fis_\markup { \dynamic rf }
bes^\markup {
  \beam #8 #0.1 #0.5
}
cis
d-\markup {
  \markalphabet #8
  \markletter #8
}
```



Più in generale, qualsiasi simbolo musicale disponibile può essere incluso separatamente in un oggetto markup, come è illustrato sotto. Una lista completa di questi simboli e dei loro nomi si trova in Sezione A.8 [The Feta font], pagina 632.

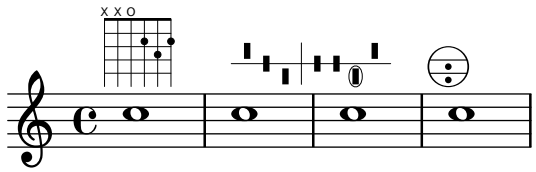
```
c2
c'^\markup { \musicglyph #"eight" }
c,4
c,8._\markup { \musicglyph #"clefs.G_change" }
c16
c2^\markup { \musicglyph #"timesig.neomensural94" }
```



Un altro modo per stampare glifi non testuali è descritto in ⟨undefined⟩ [Fonts explained], pagina ⟨undefined⟩. È utile per stampare parentesi di varie dimensioni.

La modalità markup supporta anche i diagrammi per strumenti specifici:

```
c1^\markup {
  \fret-diagram-terse #"x;x;o;2;3;2;"
}
c^\markup {
  \harp-pedal #"^-v|--ov^"
}
c
c^\markup {
  \combine
    \musicglyph #"accordion.discant"
    \combine
      \raise #0.5 \musicglyph #"accordion.dot"
      \raise #1.5 \musicglyph #"accordion.dot"
}
```
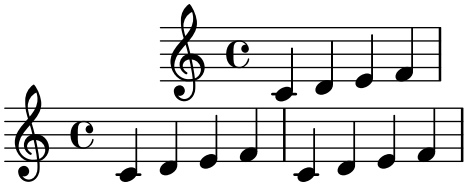


Questi diagrammi sono documentati in Sezione A.10.5 [Instrument Specific Markup], pagina 689.

È possibile annidare perfino un'intera partitura in un oggetto markup. In tal caso, il blocco `\score` annidato deve contenere un blocco `\layout`, come è illustrato qui:

```
c4 d^\markup {
  \score {
    \relative c' { c4 d e f }
    \layout { }
  }
```

```
}
e f |
c d e f
```



Una lista completa dei comandi relativi alla notazione musicale si trova in Sezione A.10.4 [Music], pagina 684.

Vedi anche

Guida alla notazione: Sezione A.10.4 [Music], pagina 684, Sezione A.8 [The Feta font], pagina 632, ⟨undefined⟩ [Fonts explained], pagina ⟨undefined⟩.

File installati: ‘`scm/define-markup-commands.scm`’, ‘`scm/fret-diagrams.scm`’, ‘`scm/harp-pedals.scm`’.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

Testo formattato su più pagine

Sebbene gli oggetti di markup standard non possano avere interruzioni, una specifica sintassi permette di inserire linee di testo che possono estendersi per varie pagine:

```
\markuplist {
  \justified-lines {
    Un testo molto lungo di linee giustificate.
    ...
  }
  \wordwrap-lines {
    Un altro paragrafo molto lungo.
    ...
  }
  ...
}
```

Un testo molto lungo di linee giustificate. ...

Un altro paragrafo molto lungo. ...

...

Questa sintassi accetta una lista di oggetti di markup, che possono essere

- il risultato di un comando \markuplist,
- una lista di markup,
- una lista di \markuplists.

Una lista completa dei comandi che si possono usare con \markuplist si trova in Sezione A.11 [Text markup list commands], pagina 703.

Vedi anche

Guida alla notazione: Sezione A.11 [Text markup list commands], pagina 703.

Estendere LilyPond: Sezione “New markup list command definition” in *Estendere*.

File installati: ‘`scm/define-markup-commands.scm`’.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

Comandi predefiniti

`\markuplist`.

1.8.3 Tipi di carattere

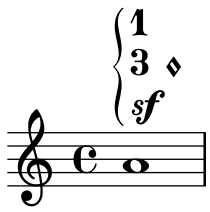
Questa sezione presenta il modo in cui sono gestiti i tipi di carattere e come possono essere modificati nelle partiture.

Tipi di carattere in dettaglio

I tipi di carattere vengono gestiti attraverso varie librerie. FontConfig rileva i tipi di carattere disponibili nel sistema; i tipi selezionati sono riprodotti con Pango.

I tipi di carattere della notazione musicale possono essere descritti come un insieme di glifi specifici, ordinati in varie famiglie. La seguente sintassi permette di usare vari caratteri `feta` di LilyPond (non testuali) direttamente nella modalità markup:

```
a1^\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup #"brace120"
    \override #'(font-encoding . fetaText)
    \column { 1 3 sf }
    \override #'(font-encoding . fetaMusic)
    \lookup #"noteheads.s0petrucci"
  }
}
```



Tuttavia, tutti questi glifi, ad eccezione delle graffe di varie dimensioni contenute in `fetaBraces`, sono già utilizzabili con la sintassi ben più semplice descritta in ⟨undefined⟩ [Music notation inside markup], pagina ⟨undefined⟩.
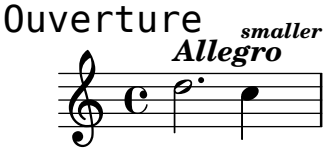
Quando si usano i glifi contenuti in `fetaBraces`, la dimensione della graffa viene specificata dalla parte numerica del nome del glifo, in unità arbitrarie. Può essere specificato qualsiasi numero intero da `0` a `575` compresi, dove `0` corrisponde alla graffa più piccola. Il valore ottimale deve essere determinato per tentativi. Questi glifi sono tutte graffe sinistre; le graffe destre si possono ottenere con la rotazione, vedi Sezione 5.4.8 [Rotating objects], pagina 598.

Sono disponibili tre famiglie di tipi di carattere: il *roman* (con grazie), che usa di default New Century Schoolbook, il *sans* (senza grazie) e il tipo monospaziato *typewriter* – queste ultime due famiglie sono determinate dall’installazione di Pango.

Nota: Non ci sono tipi predefiniti associati con le famiglie *sans* e *typewriter*. Un file di input che usa una di queste famiglie può produrre output diversi su computer diversi. Per garantire un output coerente su piattaforme diverse, occorre specificare i tipi di carattere per nome e quei tipi devono essere presenti in qualsiasi sistema che elabori il file. Si veda ⟨undefined⟩ [Single entry fonts], pagina ⟨undefined⟩ e ⟨undefined⟩ [Entire document fonts], pagina ⟨undefined⟩.

Ogni famiglia può avere forme e serie differenti. L'esempio seguente illustra la possibilità di scegliere famiglie, forme, serie e dimensioni alternative. Il valore specificato per `font-size` è la modifica relativa alla dimensione predefinita.

```
\override Score.RehearsalMark.font-family = #'typewriter
\mark \markup "Ouverture"
\override Voice.TextScript.font-shape = #'italic
\override Voice.TextScript.font-series = #'bold
d2.^\markup "Allegro"
\override Voice.TextScript.font-size = #-3
c4^smaller
```



Una sintassi simile si usa nella modalità markup; tuttavia in questo caso è preferibile usare la sintassi più semplice spiegata in ⟨undefined⟩ [Selecting font and font size], pagina ⟨undefined⟩:

```
\markup {
  \column {
    \line {
      \override #'(font-shape . italic)
      \override #'(font-size . 4)
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter)
      {
        \override #'(font-series . bold)
        re
        di
      }
      \override #'(font-family . sans)
      Creta
    }
  }
}
```



Sebbene sia semplice passare a un tipo di carattere preconfigurato, è anche possibile usare altri tipi, come viene spiegato nelle sezioni successive: ⟨undefined⟩ [Single entry fonts], pagina ⟨undefined⟩ e ⟨undefined⟩ [Entire document fonts], pagina ⟨undefined⟩.

Vedi anche
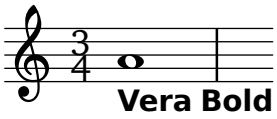
Guida alla notazione: Sezione A.8 [The Feta font], pagina 632, ⟨undefined⟩ [Music notation inside markup], pagina ⟨undefined⟩, Sezione 5.4.8 [Rotating objects], pagina 598, ⟨undefined⟩ [Selecting font and font size], pagina ⟨undefined⟩, Sezione A.10.1 [Font], pagina 653.

Tipi di carattere per singolo oggetto

Si può usare nella partitura qualsiasi tipo di carattere che sia installato nel sistema operativo e riconosciuto da FontConfig, usando la seguente sintassi:

```
\override Staff.TimeSignature.font-name = #"Bitstream Charter"
\override Staff.TimeSignature.font-size = #2
\time 3/4

a1_\markup {
  \override #'(font-name . "Vera Bold")
    { Vera Bold }
}
```



Il seguente comando mostra un elenco di tutti i tipi di carattere disponibili nel sistema operativo:

```
lilypond -dshow-available-fonts x
```

Vedi anche

Guida alla notazione: ⟨undefined⟩ [Fonts explained], pagina ⟨undefined⟩, ⟨undefined⟩ [Entire document fonts], pagina ⟨undefined⟩.

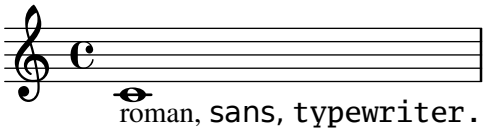
Frammenti: Sezione “Text” in *Frammenti di codice*.

Tipi di carattere per l’intero documento

È possibile modificare i tipi di carattere usati come tipi predefiniti nelle famiglie *roman*, *sans* e *typewriter* specificandoli, in questo ordine, come è mostrato nell’esempio seguente, che ridimensiona automaticamente i caratteri col valore impostato per la dimensione globale del rigo. I tipi di carattere sono spiegati in ⟨undefined⟩ [Fonts explained], pagina ⟨undefined⟩.

```
\paper  {
  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans"
                          "Luxi Mono"
                          (/ staff-height pt 20)))
}

\relative c'{
  c1-\markup {
    roman,
    \sans sans,
    \typewriter typewriter. }
}
```



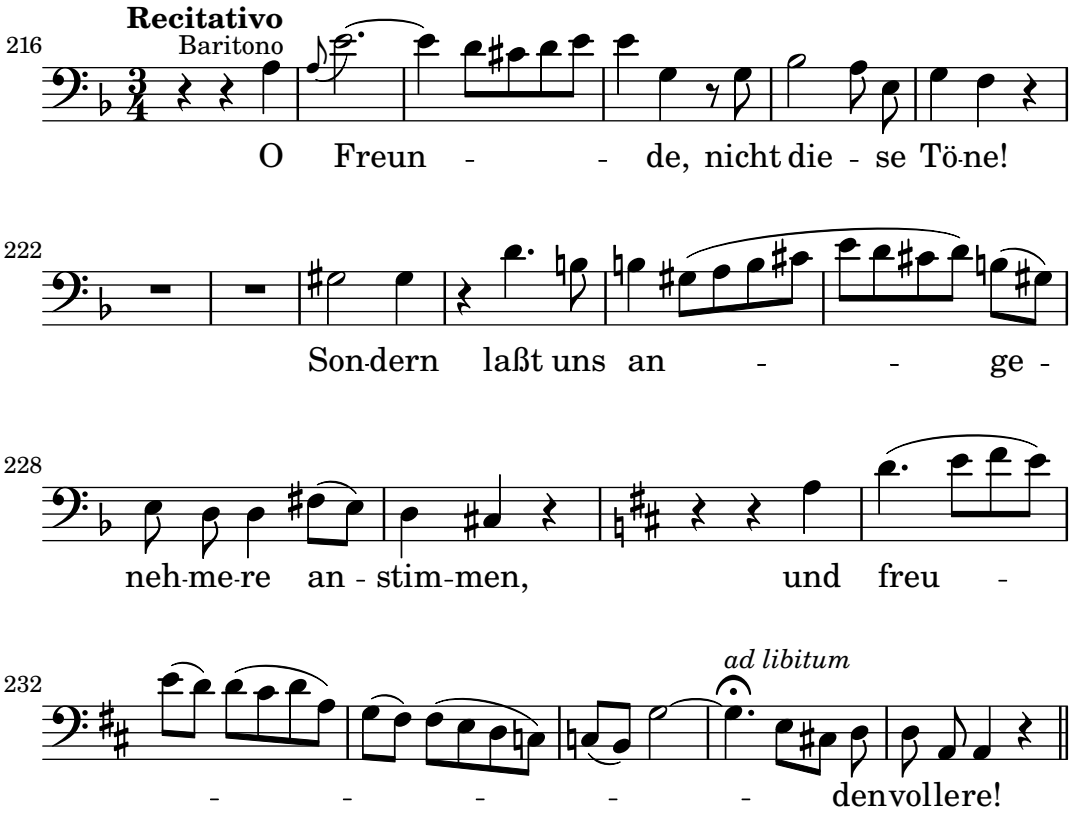
Vedi anche

Guida alla notazione: ⟨undefined⟩ [Fonts explained], pagina ⟨undefined⟩, ⟨undefined⟩ [Single entry fonts], pagina ⟨undefined⟩, ⟨undefined⟩ [Selecting font and font size], pagina ⟨undefined⟩, Sezione A.10.1 [Font], pagina 653.

2 Specialist notation

This chapter explains how to create musical notation for specific types of instrument or in specific styles.

2.1 Vocal music



This section explains how to typeset vocal music, and make sure that the lyrics will be aligned with the notes of their melody.

2.1.1 Common notation for vocal music

This section discusses issues common to most types of vocal music.

References for vocal music

This section indicates where to find details of notation issues that may arise in any type of vocal music.

- Most styles of vocal music use written text as lyrics. An introduction to this notation is to be found in Sezione “Setting simple songs” in *Manuale di Apprendimento*.
- Vocal music is likely to require the use of `markup` mode, either for lyrics or for other text elements (characters’ names, etc.) This syntax is described in ⟨undefined⟩ [Text markup introduction], pagina ⟨undefined⟩.
- *Ambitus* may be added at the beginning of vocal staves, as explained in [Ambitus], pagina 33.
- Dynamic markings by default are placed below the staff, but in choral music they are usually placed above the staff in order to avoid the lyrics, as explained in [Score layouts for choral], pagina 287.

Vedi anche

Music Glossary: Sezione “ambitus” in *Glossario Musicale*.

Learning Manual: Sezione “Setting simple songs” in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Text markup introduction], pagina ⟨undefined⟩, [Ambitus], pagina 33, [Score layouts for choral], pagina 287.

Snippets: Sezione “Vocal music” in *Frammenti di codice*.

Entering lyrics

Lyrics are entered in a special input mode, which can be introduced by the keyword `\lyricmode`, or by using `\addlyrics` or `\lyricsto`. In this special input mode, the input `d` is not parsed as the pitch *D*, but rather as a one-letter syllable of text. In other words, syllables are entered like notes but with pitches replaced by text.

For example:

```
\lyricmode { Three4 blind mice,2 three4 blind mice2 }
```

There are two main methods for specifying the horizontal placement of the syllables, either by specifying the duration of each syllable explicitly, as in the example above, or by leaving the lyrics to be aligned automatically to a melody or other voice of music, using `\addlyrics` or `\lyricsto`. The former method is described below in [Manual syllable durations], pagina 253. The latter method is described in [Automatic syllable durations], pagina 251.

A word or syllable of lyrics begins with an alphabetic character (plus some other characters, see below) and is terminated by any white space or a digit. Later characters in the syllable can be any character that is not a digit or white space.

Because any character that is not a digit or white space is regarded as part of the syllable, a word is valid even if it ends with `}`, which often leads to the following mistake:
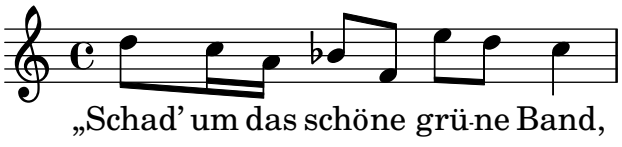
```
\lyricmode { lah lah lah}
```

In this example, the `}` is included in the final syllable, so the opening brace is not balanced and the input file will probably not compile. Instead, braces should always be surrounded with white space:

```
\lyricmode { lah lah lah }
```
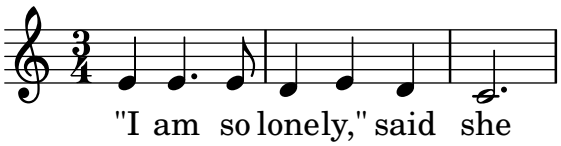
Punctuation, lyrics with accented characters, characters from non-English languages, or special characters (such as the heart symbol or slanted quotes), may simply be inserted directly into the input file, providing it is saved with UTF-8 encoding. For more information, see Sezione 3.3.3 [Special characters], pagina 486.

```
\relative c'' { d8 c16 a bes8 f e' d c4 }
\addlyrics { „Schad’ um das schö -- ne grü -- ne Band, }
```



Normal quotes may be used in lyrics, but they have to be preceded with a backslash character and the whole syllable has to be enclosed between additional quotes. For example,

```
\relative c' { \time 3/4 e4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone -- "ly,\"" said she }
```



The full definition of a word start in lyrics mode is somewhat more complex. A word in lyrics mode is one that begins with an alphabetic character, `_`, `?`, `!`, `:`, `'`, the control characters `^A` through `^F`, `^Q` through `^W`, `^Y`, `^^`, any 8-bit character with an ASCII code over 127, or a two-character combination of a backslash followed by one of `` ` ``, `'`, `"`, or `^`.

Great control over the appearance of lyrics comes from using `\markup` inside the lyrics themselves. For explanation of many options, see ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Frammenti di codice selezionati

Formatting lyrics syllables

Markup mode may be used to format individual syllables in lyrics.

```
mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>
```



Vedi anche

Learning Manual: Sezione “Songs” in *Manuale di Apprendimento*.

Notation Reference: [Automatic syllable durations], pagina 251, ⟨undefined⟩ [Fonts], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, Sezione 5.4.1 [Input modes], pagina 583, [Manual syllable durations], pagina 253, Sezione 3.3.3 [Special characters], pagina 486.

Internals Reference: Sezione “LyricText” in *Guida al Funzionamento Interno*.

Snippets: Sezione “Text” in *Frammenti di codice*.

Aligning lyrics to a melody

Lyrics are printed by interpreting them in the context called `Lyrics`, see Sezione 5.1.1 [Contexts explained], pagina 553.

```
\new Lyrics \lyricmode { ... }
```

Lyrics can be aligned with melodies in two main ways:

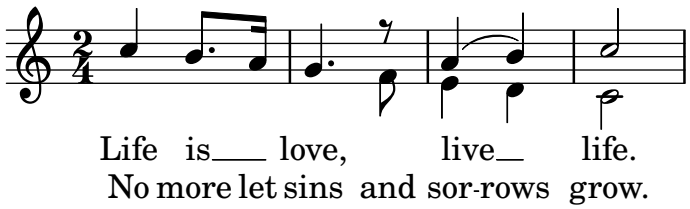
- Lyrics can be aligned automatically, with the durations of the syllables being taken from another voice of music or (in special circumstances) an associated melody, using `\addlyrics`, `\lyricsto`, or by setting the `associatedVoice` property. For more details, see [Automatic syllable durations], pagina 251.

  ```
  <<
    \new Staff <<
      \time 2/4
  ```

```
    \new Voice = "one" \relative c'' {
      \voiceOne
      c4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative c' {
      \voiceTwo
       s2 s4. f8 e4 d c2
    }
  >>

% takes durations and alignment from notes in "one"
  \new Lyrics \lyricsto "one" {
    Life is __ _ love, live __ life.
  }

% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



The first stanza shows the normal way of entering lyrics.

The second stanza shows how the voice from which the lyric durations are taken can be changed. This is useful if the words to different stanzas fit the notes in different ways and all the durations are available in Voice contexts. For more details, see Sezione 2.1.3 [Stanzas], pagina 278.
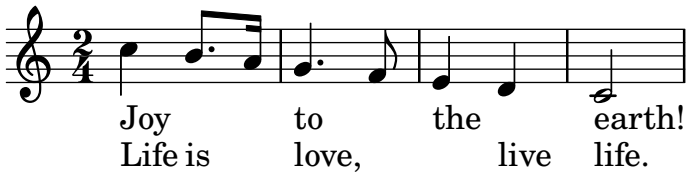
- Lyrics can be aligned independently of the duration of any notes if the durations of the syllables are specified explicitly, and entered with `\lyricmode`.

```
<<
  \new Voice = "one" \relative c'' {
    \time 2/4
    c4 b8. a16 g4. f8 e4 d c2
  }

% uses previous explicit duration of 2;
  \new Lyrics \lyricmode {
    Joy to the earth!
  }

% explicit durations, set to a different rhythm
  \new Lyrics \lyricmode {
    Life4 is love,2. live4 life.2
```

```
  }
>>
```



The first stanza is not aligned with the notes because the durations were not specified, and the previous value of 2 is used for each word.

The second stanza shows how the words can be aligned quite independently from the notes. This is useful if the words to different stanzas fit the notes in different ways and the required durations are not available in a music context. For more details see [Manual syllable durations], pagina 253. This technique is also useful when setting dialogue over music; for examples showing this, see [Dialogue over music], pagina 296.

When entered in this way the words are left-aligned to the notes by default, but may be center-aligned to the notes of a melody by specifying an associated voice, if one exists. For details, see [Manual syllable durations], pagina 253.

Vedi anche

Learning Manual: Sezione “Aligning lyrics to a melody” in *Manuale di Apprendimento*.

Notation Reference: Sezione 5.1.1 [Contexts explained], pagina 553, [Automatic syllable durations], pagina 251, Sezione 2.1.3 [Stanzas], pagina 278, [Manual syllable durations], pagina 253, [Dialogue over music], pagina 296, [Manual syllable durations], pagina 253.

Internals Reference: Sezione “Lyrics” in *Guida al Funzionamento Interno*.

Automatic syllable durations

Lyrics can be automatically aligned to the notes of a melody in three ways:

- by specifying the named Voice context containing the melody with `\lyricsto`,
- by introducing the lyrics with `\addlyrics` and placing them immediately after the Voice context containing the melody,
- by setting the `associatedVoice` property, the alignment of the lyrics may be switched to a different named Voice context at any musical moment.

In all three methods hyphens can be drawn between the syllables of a word and extender lines can be drawn beyond the end of a word. For details, see [Extenders and hyphens], pagina 259.
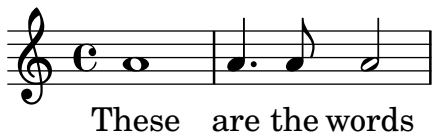
The `Voice` context containing the melody to which the lyrics are being aligned must not have “died”, or the lyrics after that point will be lost. This can happen if there are periods when that voice has nothing to do. For methods of keeping contexts alive, see Sezione 5.1.3 [Keeping contexts alive], pagina 558.

Using `\lyricsto`

Lyrics can be aligned under a melody automatically by specifying the named Voice context containing the melody with `\lyricsto`:

```
<<
  \new Voice = "melody" {
    a1 a4. a8 a2
  }
  \new Lyrics \lyricsto "melody" {
```

```
    These are the words
  }
>>
```



This aligns the lyrics to the notes of the named `Voice` context, which must already exist. Therefore normally the `Voice` context is specified first, followed by the `Lyrics` context. The lyrics themselves follow the `\lyricsto` command. The `\lyricsto` command invokes lyric mode automatically, so the `\lyricmode` keyword may be omitted. By default, the lyrics are placed underneath the notes. For other placements, see [Placing lyrics vertically], pagina 261.

Using `\addlyrics`

The `\addlyrics` command is just a convenient shortcut that can sometimes be used instead of having to set up the lyrics through a more complicated LilyPond structure.
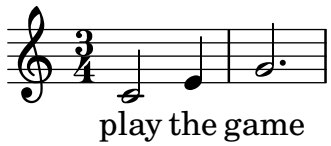
```
{ MUSIC }
\addlyrics { LYRICS }
```

is the same as

```
\new Voice = "blah" { MUSIC }
\new Lyrics \lyricsto "blah" { LYRICS }
```
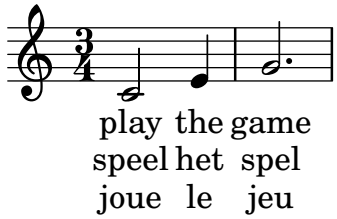
Here is an example,

```
{
  \time 3/4
  \relative c' { c2 e4 g2. }
  \addlyrics { play the game }
}
```



More stanzas can be added by adding more `\addlyrics` sections:

```
{
  \time 3/4
  \relative c' { c2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
}
```



The command `\addlyrics` cannot handle polyphonic settings. Also, it cannot be used to associate lyrics to a `TabVoice`. For these cases one should use `\lyricsto`.

Using associatedVoice
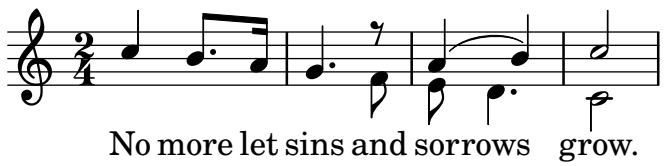
The melody to which the lyrics are being aligned can be changed by setting the `associatedVoice` property,

```
\set associatedVoice = #"lala"
```

The value of the property (here: `"lala"`) should be the name of a `Voice` context. For technical reasons, the `\set` command must be placed one syllable before the one to which the change in voice is to apply.

Here is an example demonstrating its use:

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative c'' {
      \voiceOne
      c4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative c' {
      \voiceTwo
       s2 s4. f8 e8 d4. c2
    }
  >>
% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



Vedi anche

Notation Reference: [Extenders and hyphens], pagina 259, Sezione 5.1.3 [Keeping contexts alive], pagina 558, [Placing lyrics vertically], pagina 261.

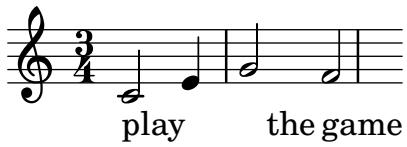
Manual syllable durations

In some complex vocal music, it may be desirable to place lyrics completely independently of notes. In this case do not use `\lyricsto` or `\addlyrics` and do not set `associatedVoice`. Syllables are entered like notes – but with pitches replaced by text – and the duration of each syllable is entered explicitly after the syllable.

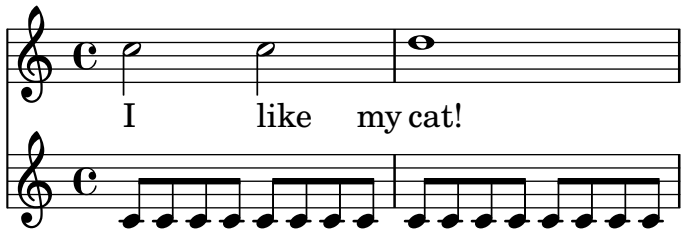
By default, syllables will be left-aligned to the corresponding musical moment. Hyphenated lines may be drawn between syllables as usual, but extender lines cannot be drawn when there is no associated voice.

Here are two examples:

```
<<
  \new Voice = "melody" {
    \time 3/4
    c2 e4 g2 f
  }
  \new Lyrics \lyricmode {
    play1 the4 game4
  }
>>
```




```
<<
  \new Staff {
    \relative c'' {
      c2 c2
      d1
    }
  }
  \new Lyrics {
    \lyricmode {
      I2 like4. my8 cat!1
    }
  }
  \new Staff {
    \relative c' {
      c8 c c c c c c c
      c8 c c c c c c c
    }
  }
>>
```
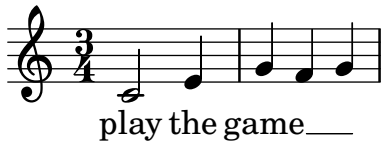


This technique is useful when writing dialogue over music, see [Dialogue over music], pagina 296.

To center-align syllables on the notes at the corresponding musical moments, set `associatedVoice` to the name of the Voice context containing those notes. When `associatedVoice` is set, both double hyphens and double underscores can be used to draw hyphenated lines and extenders under melismata correctly.

```
<<
  \new Voice = "melody" {
    \time 3/4
```

```
    c2 e4 g f g
  }
  \new Lyrics \lyricmode {
    \set associatedVoice = #"melody"
    play2 the4 game2. __
  }
>>
```



Vedi anche

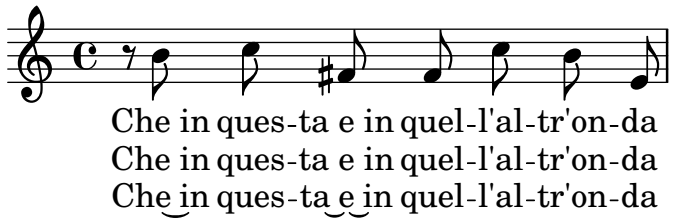
Notation Reference: [Dialogue over music], pagina 296.

Internals Reference: Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

Multiple syllables to one note

In order to assign more than one syllable to a single note with spaces between the syllables, you can surround the phrase with quotes or use a _ character. Alternatively, you can use the tilde symbol (~) to get a lyric tie.

```
{
  { \autoBeamOff
    r8 b c fis, fis c' b e,
  }
  \addlyrics
  {
    \override LyricHyphen.minimum-distance = #1.0 % Ensure hyphens are visible
    Che_in ques -- ta_e_in quel -- l'al -- tr'on -- da
  }
  \addlyrics { "Che in" ques -- "ta e in" quel -- l'al -- tr'on -- da }
  \addlyrics { Che~in ques -- ta~e~in quel -- l'al -- tr'on -- da }
}
```



Vedi anche

Internals Reference: Sezione “LyricCombineMusic” in *Guida al Funzionamento Interno*.

Multiple notes to one syllable

Sometimes, particularly in Medieval and baroque music, several notes are sung on one syllable; this is called melisma, see Sezione “melisma” in *Glossario Musicale*. The syllable to a melisma is usually left-aligned with the first note of the melisma.
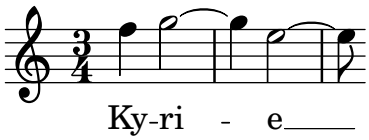
When a melisma occurs on a syllable other than the last one in a word, that syllable is usually joined to the following one with a hyphenated line. This is indicated by placing a double hyphen, `--`, immediately after the syllable.

Alternatively, when a melisma occurs on the last or only syllable in a word an extender line is usually drawn from the end of the syllable to the last note of the melisma. This is indicated by placing a double underscore, `__`, immediately after the word.

There are five ways in which melismata can be indicated:
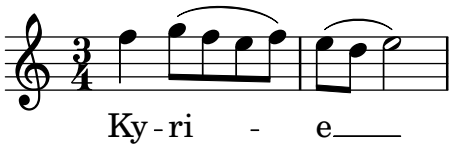
- Melismata are created automatically over notes which are tied together:

```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g2 ~ |
    g4 e2 ~ |
    e8
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```



- Melismata can be created automatically from the music by placing slurs over the notes of each melisma. This is the usual way of entering lyrics:

```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g8 ( f e f )
    e8 ( d e2 )
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```
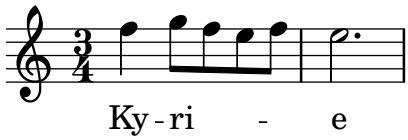


  Note that phrasing slurs do not affect the creation of melismata.

- Notes are considered a melisma if they are manually beamed, providing automatic beaming is switched off. See ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩.

```
<<
  \new Voice = "melody" {
    \time 3/4
    \autoBeamOff
    f4 g8[ f e f]
```
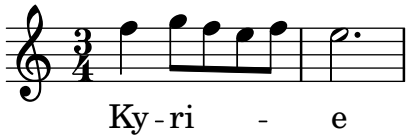
```
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```



Clearly this is not suited to melismata over notes which are longer than eighth notes.
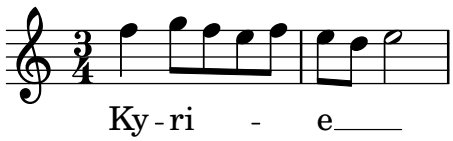
- An unslurred group of notes will be treated as a melisma if they are bracketed between `\melisma` and `\melismaEnd`.

```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g8
    \melisma
    f e f
    \melismaEnd
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```
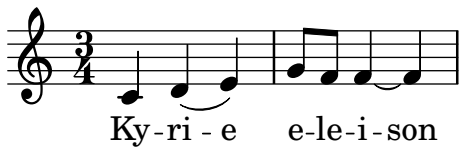


- A melisma can be defined entirely in the lyrics by entering a single underscore character, `_`, for every extra note that has to be added to the melisma.

```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g8 f e f
    e8 d e2
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ _ _ e __ _ _
  }
>>
```



It is possible to have ties, slurs and manual beams in the melody without their indicating melismata. To do this, set `melismaBusyProperties`:

```
<<
  \new Voice = "melody" {
    \time 3/4
    \set melismaBusyProperties = #'()
    c4 d ( e )
    g8 [ f ] f4 ~ f
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e e -- le -- i -- son
  }
>>
```
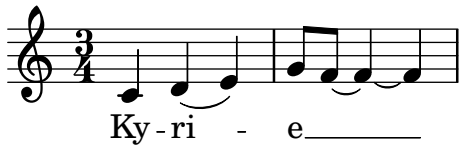


Other settings for `melismaBusyProperties` can be used to selectively include or exclude ties, slurs, and beams from the automatic detection of melismata; see `melismaBusyProperties` in Sezione "Tunable context properties" in *Guida al Funzionamento Interno*.

Alternatively, if all melismata indications are to be ignored, `ignoreMelismata` may be set true; see [Stanzas with different rhythms], pagina 279.

If a melisma is required during a passage in which `melismaBusyProperties` is active, it may be indicated by placing a single underscore in the lyrics for each note which should be included in the melisma:

```
<<
  \new Voice = "melody" {
    \time 3/4
    \set melismaBusyProperties = #'()
    c4 d ( e )
    g8 [ f ] ~ f4 ~ f
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ e __ _ _ _
  }
>>
```



Comandi predefiniti

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

Vedi anche

Musical Glossary: Sezione "melisma" in *Glossario Musicale*.

Learning Manual: Sezione "Aligning lyrics to a melody" in *Manuale di Apprendimento*.

Notation Reference: [Aligning lyrics to a melody], pagina 249, [Automatic syllable durations], pagina 251, ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩, [Stanzas with different rhythms], pagina 279.

Internals Reference: Sezione “Tunable context properties” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Extender lines under melismata are not created automatically; they must be inserted manually with a double underscore.

Extenders and hyphens

In the last syllable of a word, melismata are sometimes indicated with a long horizontal line starting in the melisma syllable, and ending in the next one. Such a line is called an extender line, and it is entered as ‘ `__` ’ (note the spaces before and after the two underscore characters).

Nota: Melismata are indicated in the score with extender lines, which are entered as one double underscore; but short melismata can also be entered by skipping individual notes, which are entered as single underscore characters; these do not make an extender line to be typeset by default.

Centered hyphens are entered as ‘ `--` ’ between syllables of a same word (note the spaces before and after the two hyphen characters). The hyphen will be centered between the syllables, and its length will be adjusted depending on the space between the syllables.

In tightly engraved music, hyphens can be removed. Whether this happens can be controlled with the `minimum-distance` (minimum distance between two syllables) and the `minimum-length` (threshold below which hyphens are removed) properties of `LyricHyphen`.

Vedi anche
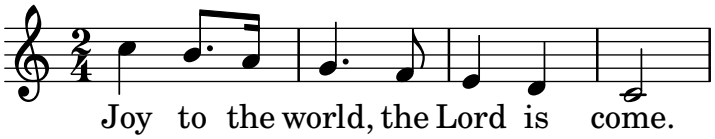
Internals Reference: Sezione “LyricExtender” in *Guida al Funzionamento Interno*, Sezione “LyricHyphen” in *Guida al Funzionamento Interno*.

2.1.2 Techniques specific to lyrics

Working with lyrics and variables

Variables containing lyrics can be created, but the lyrics must be entered in lyric mode:

```
musicOne = \relative c'' {
  c4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  <<
    \new Voice = "one" {
      \time 2/4
      \musicOne
    }
    \new Lyrics \lyricsto "one" {
      \verseOne
    }
  >>
}
```
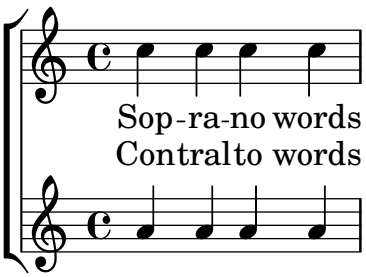


Durations do not need to be added if the variable is to be invoked with `\addlyrics` or `\lyricsto`.

For different or more complex orderings, the best way is to define the music and lyric variables first, then set up the hierarchy of staves and lyrics, omitting the lyrics themselves, and then add the lyrics using `\context` underneath. This ensures that the voices referenced by `\lyricsto` have always been defined earlier. For example:

```
sopranoMusic = \relative c'' { c4 c c c }
contraltoMusic = \relative c'' { a4 a a a }
sopranoWords = \lyricmode { Sop -- ra -- no words }
contraltoWords = \lyricmode { Con -- tral -- to words }

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \sopranoMusic
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos"
    \new Staff {
      \new Voice = "contraltos" {
        \contraltoMusic
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        \sopranoWords
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        \contraltoWords
      }
    }
  >>
}
```



Vedi anche

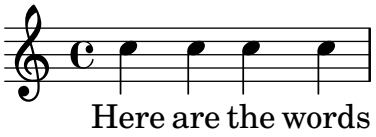
Notation Reference: [Placing lyrics vertically], pagina 261.

Internals Reference: Sezione “LyricCombineMusic” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*.

Placing lyrics vertically

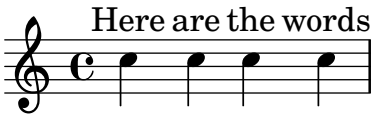
Depending on the type of music, lyrics may be positioned above the staff, below the staff, or between staves. Placing lyrics below the associated staff is the easiest, and can be achieved by simply defining the Lyrics context below the Staff context:

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' { c4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



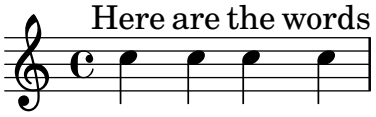
Lyrics may be positioned above the staff using one of two methods. The simplest (and preferred) method is to use the same syntax as above and explicitly specify the position of the lyrics:

```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative c'' { c4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Alternatively, a two-step process may be used. First the Lyrics context is declared (without any content) before the Staff and Voice contexts, then the `\lyricsto` command is placed after the Voice declaration it references by using `\context`, as follows:

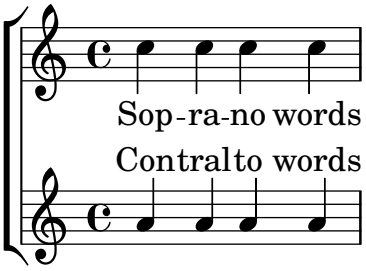
```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "melody" {
        \relative c'' { c4 c c c }
      }
    }
    \context Lyrics = "lyrics" {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



When there are two voices on separate staves the lyrics may be placed between the staves using either of these methods. Here is an example of the second method:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative c'' { c4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "contraltos" {
        \relative c'' { a4 a a a }
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        Sop -- ra -- no words
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        Con -- tral -- to words
      }
    }
```

```
  >>
}
```



Other combinations of lyrics and staves may be generated by elaborating these examples, or by examining the templates in the Learning Manual, see Sezione “Vocal ensembles templates” in *Manuale di Apprendimento*.

Frammenti di codice selezionati

Obtaining 2.12 lyrics spacing in newer versions

The vertical spacing engine changed for version 2.14. This can cause lyrics to be spaced differently. It is possible to set properties for `Lyric` and `Staff` contexts to get the spacing engine to behave as it did in version 2.12.

```
global = {
  \key d \major
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

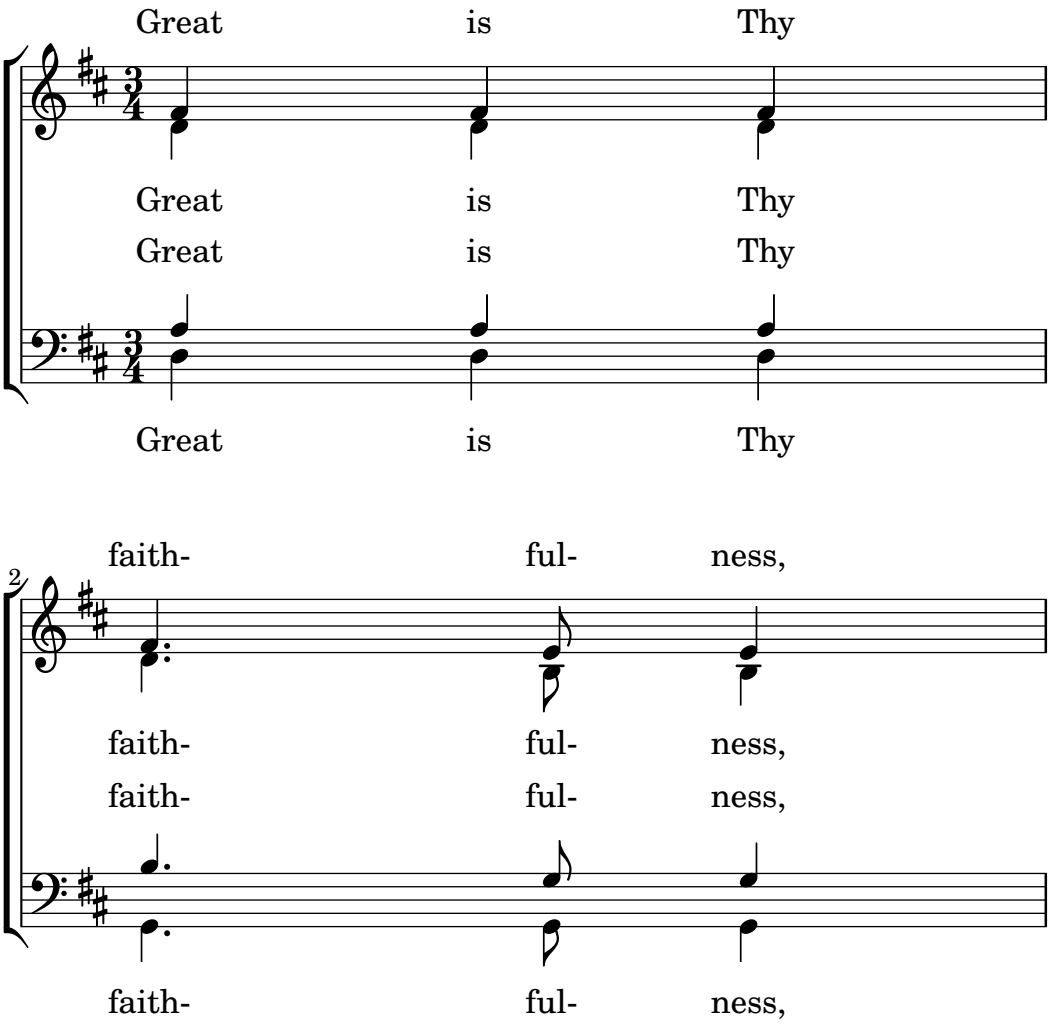
altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
}

bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}

words = \lyricmode {
  Great is Thy faith- ful- ness,
}
```

```
\score {
  \new ChoirStaff <<
    \new Lyrics = sopranos
    \new Staff = women <<
      \new Voice = "sopranos" {
        \voiceOne
        \global \sopMusic
      }
      \new Voice = "altos" {
        \voiceTwo
        \global \altoMusic
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors"
    \new Staff = men <<
      \clef bass
      \new Voice = "tenors" {
        \voiceOne
        \global \tenorMusic
      }
      \new Voice = "basses" {
        \voiceTwo  \global \bassMusic
      }
    >>
    \new Lyrics = basses
    \context Lyrics = sopranos \lyricsto sopranos \words
    \context Lyrics = altos \lyricsto altos \words
    \context Lyrics = tenors \lyricsto tenors \words
    \context Lyrics = basses \lyricsto basses \words
  >>
  \layout {
    \context {
      \Lyrics
      \override VerticalAxisGroup.staff-affinity = ##f
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
    \context {
      \Staff
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
  }
}
```



Vedi anche
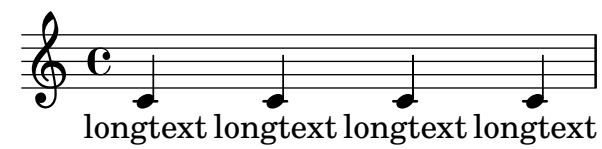
Learning Manual: Sezione “Vocal ensembles templates” in *Manuale di Apprendimento*.
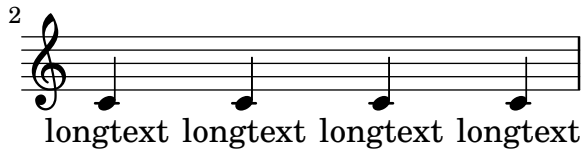
Notation Reference: Sezione 5.1.7 [Context layout order], pagina 570, Sezione 5.1.2 [Creating and referencing contexts], pagina 555.

Placing syllables horizontally

To increase the spacing between lyrics, set the `minimum-distance` property of `LyricSpace`.

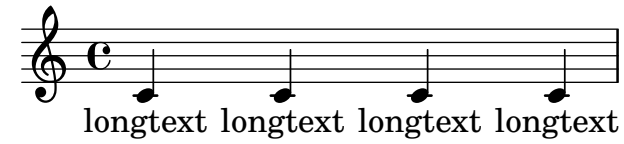
```
{
  c c c c
  \override Lyrics.LyricSpace.minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```
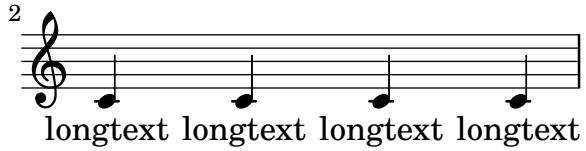




To make this change for all lyrics in the score, set the property in the `\layout` block.

```
\score {
  \relative c' {
  c c c c
  c c c c
  }
  \addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace.minimum-distance = #1.0
    }
  }
}
```
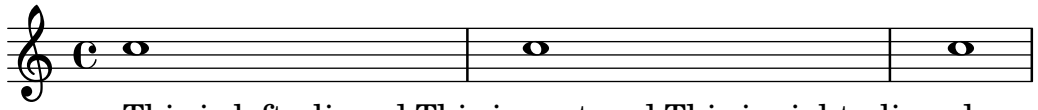




Frammenti di codice selezionati

Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the `self-alignment-X` property of the `LyricText` object. `#-1` is left, `#0` is center and `#1` is right; however, you can use `#LEFT`, `#CENTER` and `#RIGHT` as well.

```
\layout { ragged-right = ##f }
\relative c'' {
  c1
  c1
  c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "This is centered"
  \once \override LyricText.self-alignment-X = #1
  "This is right-aligned"
}
```



Checking to make sure that text scripts and lyrics are within the margins requires additional calculations. To speed up processing slightly, this feature can be disabled:

```
\override Score.PaperColumn.keep-inside-line = ##f
```

To make lyrics avoid bar lines as well, use

```
\layout {
  \context {
    \Lyrics
      \consists "Bar_engraver"
      \consists "Separating_line_group_engraver"
      \hide BarLine
  }
}
```
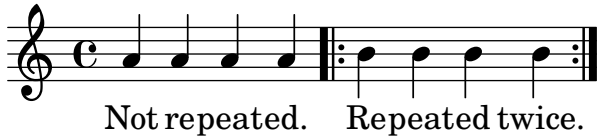
Lyrics and repeats

Simple repeats

Repeats in *music* are fully described elsewhere; see ⟨undefined⟩ [Repeats], pagina ⟨undefined⟩. This section explains how to add lyrics to repeated sections of music.

Lyrics to a section of music that is repeated should be surrounded by exactly the same repeat construct as the music, if the words are unchanged.
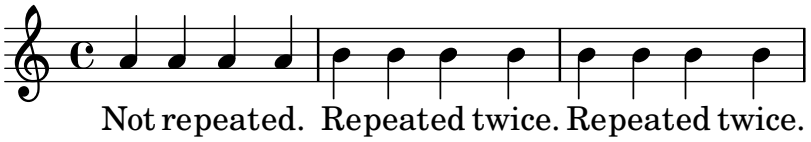
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- ed twice. }
      }
    }
  >>
}
```



The words will then be correctly expanded if the repeats are unfolded.
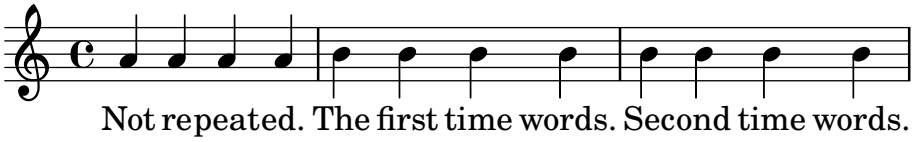
```
\score {
  \unfoldRepeats {
    <<
```

```
      \new Staff {
        \new Voice = "melody" {
          \relative c'' {
            a4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Not re -- peat -- ed.
          \repeat volta 2 { Re -- peat -- ed twice. }
        }
      }
    >>
  }
}
```
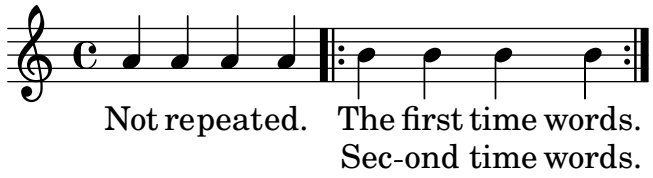


If the repeated section is to be unfolded and has different words, simply enter all the words:

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        The first time words.
        Sec -- ond time words.
      }
    }
  >>
}
```



When the words to a repeated volta section are different, the words to each repeat must be entered in separate `Lyrics` contexts, correctly nested in parallel sections:

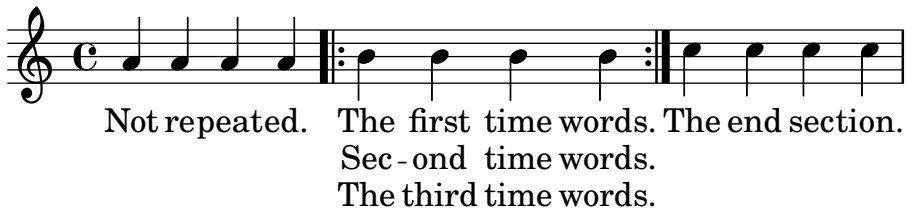
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics \lyricsto "melody" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics {
          \set associatedVoice = "melody"
          Sec -- ond time words.
        }
      >>
    }
  >>
}
```



More verses may be added in a similar way:

```
\score {
  <<
    \new Staff {
      \new Voice = "singleVoice" {
        \relative c'' {
          a4 a a a
          \repeat volta 3 { b4 b b b }
          c4 c c c
        }
      }
    }
    \new Lyrics \lyricsto "singleVoice" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics {
          \set associatedVoice = "singleVoice"
          Sec -- ond time words.
        }
        \new Lyrics {
          \set associatedVoice = "singleVoice"
          The third time words.
```

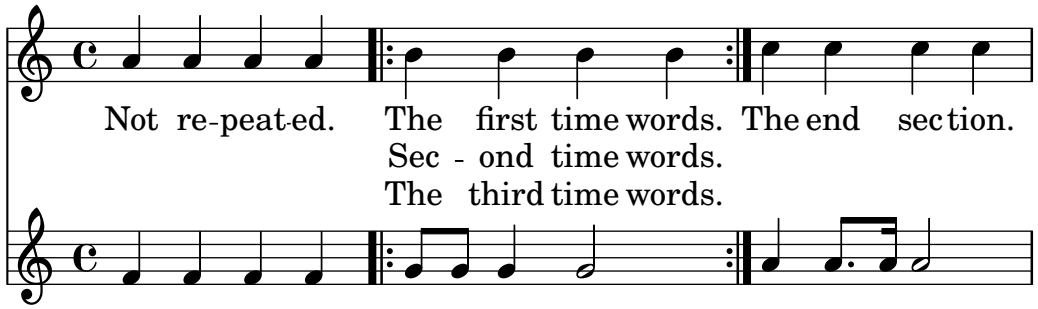
```
        }
      >>
      The end sec -- tion.
    }
  >>
}
```



However, if this construct is embedded within a multi-staved context such as a `ChoirStaff` the lyrics of the second and third verses will appear beneath the bottom staff.

To position them correctly use `alignBelowContext`:
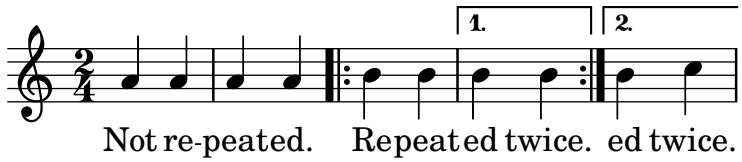
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 3 { b4 b b b }
          c4 c c c
        }
      }
    }
    \new Lyrics = "firstVerse" \lyricsto "melody" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics = "secondVerse"
        \with { alignBelowContext = #"firstVerse" } {
          \set associatedVoice = "melody"
          Sec -- ond time words.
        }
        \new Lyrics = "thirdVerse"
        \with { alignBelowContext = #"secondVerse" } {
          \set associatedVoice = "melody"
          The third time words.
        }
      >>
      The end sec -- tion.
    }
    \new Voice = "harmony" {
      \relative c' {
        f4 f f f \repeat volta 2 { g8 g g4 g2 } a4 a8. a16 a2
      }
    }
  >>
}
```



Repeats with alternative endings

If the words of the repeated section are the same, exactly the same structure can be used for both the lyrics and music.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 2 { b4 b }
          \alternative { { b b } { b c } }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- }
        \alternative { { ed twice. } { ed twice. } }
      }
    }
  >>
}
```



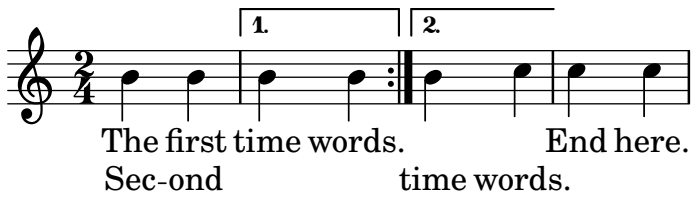
But when the repeated section has different words, a repeat construct cannot be used around the words and `\skip` commands have to be inserted manually to skip over the notes in the alternative sections which do not apply.

Note: do not use an underscore, _, to skip notes – an underscore indicates a melisma, causing the preceding syllable to be left-aligned.

Nota: The `\skip` command must be followed by a number, but this number is ignored in lyrics which derive their durations from the notes in an associated melody through `\addlyrics` or `\lyricsto`. Each `\skip` skips a single note of any value, irrespective of the value of the following number.

```
\score {
  <<
    \new Staff {
```

```
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \repeat volta 2 { b4 b }
          \alternative { { b b } { b c } }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        The first time words.
        \repeat unfold 2 { \skip 1 }
        End here.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Sec -- ond
        \repeat unfold 2 { \skip 1 }
        time words.
      }
    }
  >>
}
```

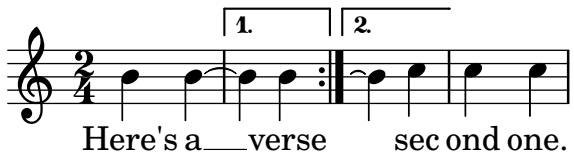


When a note is tied over into two or more alternative endings a tie is used to carry the note into the first alternative ending and a `\repeatTie` is used in the second and subsequent endings. This structure causes difficult alignment problems when lyrics are involved and increasing the length of the alternative sections so the tied notes are contained wholly within them may give a more acceptable result.

The tie creates a melisma into the first alternative, but not into the second and subsequent alternatives, so to align the lyrics correctly it is necessary to disable the automatic creation of melismata over the volta section and insert manual skips.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \set melismaBusyProperties = #'()
          \repeat volta 2 { b4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          \unset melismaBusyProperties
          c4 c
```

```
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        \repeat volta 2 { Here's a __ }
        \alternative {
          { \skip 1 verse }
          { \skip 1 sec }
        }
        ond one.
      }
    }
  >>
}
```

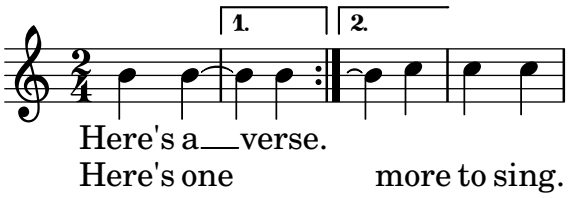


Note that if `\unfoldRepeats` is used around a section containing `\repeatTie`, the `\repeatTie` should be removed to avoid both types of tie being printed.

When the repeated section has different words a `\repeat` cannot be used around the lyrics and `\skip` commands need to be inserted manually, as before.
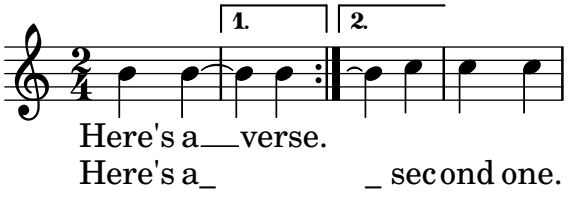
```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \repeat volta 2 { b4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's one
        \repeat unfold 2 { \skip 1 }
        more to sing.
      }
    }
  >>
```

```
}
```



If you wish to show extenders and hyphens into and out of alternative sections these must be inserted manually.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \repeat volta 2 { b4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's "a_"
        \skip 1
        "_" sec -- ond one.
      }
    }
  >>
}
```
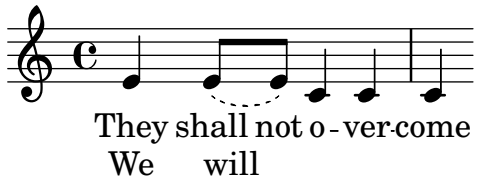


Vedi anche

Notation Reference: Sezione 5.1.3 [Keeping contexts alive], pagina 558, ⟨undefined⟩ [Repeats], pagina ⟨undefined⟩.

Divisi lyrics

When just the words and rhythms of the two parts differ with the pitches remaining the same, temporarily turning off the automatic detection of melismata and indicating the melisma in the lyrics may be the appropriate method to use:
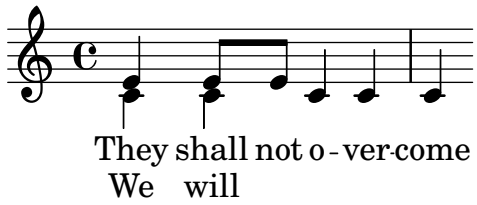
```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        \set melismaBusyProperties = #'()
        \slurDown
        \slurDashed
        e4 e8 ( e ) c4 c |
        \unset melismaBusyProperties
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "melody" {
      We will _
    }
  >>
}
```



When both music and words differ it may be better to display the differing music and lyrics by naming voice contexts and attaching lyrics to those specific contexts:

```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        <<
          {
            \voiceOne
            e4 e8 e
          }
          \new Voice = "splitpart" {
            \voiceTwo
            c4 c
          }
        >>
        \oneVoice
        c4 c |
        c
      }
```

```
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "splitpart" {
      We will
    }
  >>
}
```



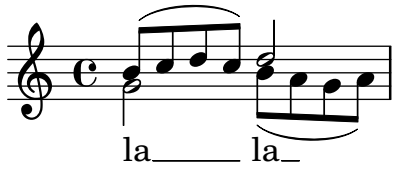
It is common in choral music to have a voice part split for several measures. The `<< {...} \\ {...} >>` construct, where the two (or more) musical expressions are separated by double backslashes, might seem the proper way to set the split voices. This construct, however, will assign **all** the expressions within it to **NEW Voice contexts** which will result in *no lyrics* being set for them since the lyrics will be set to the original voice context – not, typically, what one wants. The temporary polyphonic passage is the proper construct to use, see section *Temporary polyphonic passages* in ⟨undefined⟩ [Single-staff polyphony], pagina ⟨undefined⟩.

Polyphony with shared lyrics

When two voices with different rhythms share the same lyrics, aligning the lyrics to one of the voices may lead to problems in the other voice. For example, the second lyric extender below is too short, since the lyrics are aligned only to the top voice:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
words = \lyricmode { la __ la __ }

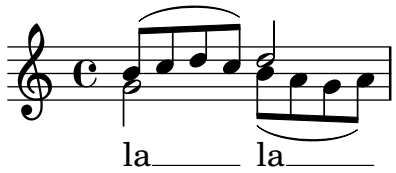
\new Staff <<
  \new Voice = "sopranoVoice" { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new Lyrics \lyricsto "sopranoVoice" \words
>>
```



To get the desired result, align the lyrics to a new `NullVoice` context containing a suitable combination of the two voices. The notes of the `NullVoice` context do not appear on the printed page, but can be used to align the lyrics appropriately:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }
```

```
\new Staff <<
  \new Voice { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
```

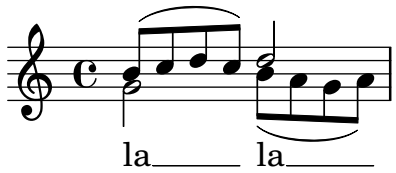


The `NullVoice` context must be placed within a `Staff` context and contain notes that are already being displayed in that staff and that are also in the same octave. Otherwise the `NullVoice` may interact with the printed voices in unexpected ways. For example, arbitrary notes in the `NullVoice` may cause accidentals to appear (or disappear) on the staff.

This method also can be used with the `\partcombine` function, which does not allow lyrics on its own:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice \partcombine \soprano \alto
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
```



Problemi noti e avvertimenti

The `\addLyrics` function only works with `Voice` lyrics and so cannot be used with `NullVoice`.

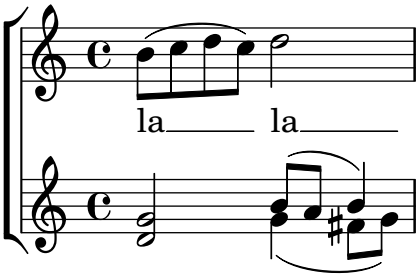
The `\partcombine` function is described in ⟨undefined⟩ [Automatic part combining], pagina ⟨undefined⟩.

Lastly, this method can be used even when the voices are in different staves, and is not limited to only two voices:

```
soprano = \relative { b'8( c d c) d2 }
altoOne = \relative { g'2 b8( a b4) }
altoTwo = \relative { d'2 g4( fis8 g) }
aligner = \relative { b'8( c d c) d( d d d) }
words = \lyricmode { la __ la __ }

\new ChoirStaff <<
  \new Staff <<
```

```
    \soprano
    \new NullVoice = "aligner" \aligner
  >>
  \new Lyrics \lyricsto "aligner" \words
  \new Staff \partcombine \altoOne \altoTwo
>>
```



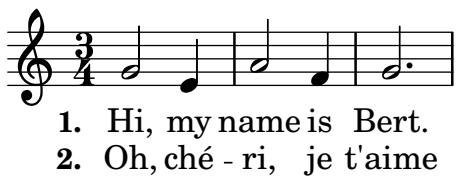
However, note that in the second half of the measure above, the notes in the `NullVoice` context reflect the rhythm of the lower staff, but they do not deviate from the single pitch being displayed in the staff to which the `NullVoice` belongs. While not actually required in this particular example, it is a good idea in general to enter the notes in this way.

2.1.3 Stanzas

Adding stanza numbers

Stanza numbers can be added by setting `stanza`, e.g.,

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = #"2. "
  Oh, ché -- ri, je t'aime
}
```



These numbers are put just before the start of the first syllable.

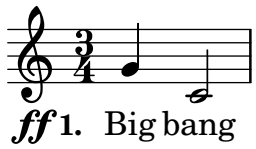
Adding dynamics marks to stanzas

Stanzas differing in loudness may be indicated by putting a dynamics mark before each stanza. In LilyPond, everything coming in front of a stanza goes into the `StanzaNumber` object; dynamics marks are no different. For technical reasons, you have to set the stanza outside `\lyricmode`:

```
text = {
  \set stanza = \markup { \dynamic "ff" "1. " }
  \lyricmode {
    Big bang
  }
```

```
}

<<
  \new Voice = "tune" {
    \time 3/4
    g'4 c'2
  }
\new Lyrics \lyricsto "tune" \text
>>
```
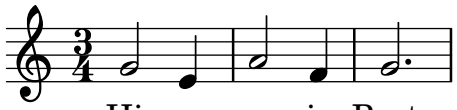


ff **1.** Big bang

Adding singers' names to stanzas

Names of singers can also be added. They are printed at the start of the line, just like instrument names. They are created by setting `vocalName`. A short version may be entered as `shortVocalName`.

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = #"Bert "
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = #"Ernie "
  Oh, ché -- ri, je t'aime
}
```



Stanzas with different rhythms

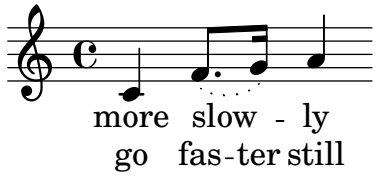
Often, different stanzas of one song are put to one melody in slightly differing ways. Such variations can still be captured with `\lyricsto`.

Ignoring melismata

One possibility is that the text has a melisma in one stanza, but multiple syllables in another. One solution is to make the faster voice ignore the melisma. This is done by setting `ignoreMelismata` in the Lyrics context.

```
<<
  \relative c' \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c4
    \slurDotted
    f8.[( g16])
    a4
  }
```

```
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    go
    \set ignoreMelismata = ##t
    fas -- ter
    \unset ignoreMelismata
    still
  }
>>
```
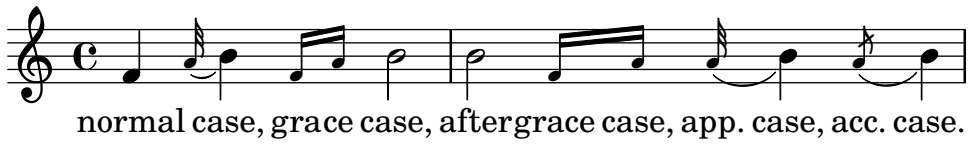


Problemi noti e avvertimenti

Unlike most `\set` commands, `\set ignoreMelismata` does not work if prefixed with `\once`. It is necessary to use `\set` and `\unset` to bracket the lyrics where melismata are to be ignored.

Adding syllables to grace notes

By default, grace notes (e.g. via `\grace`) do not get assigned syllables when using `\lyricsto`, but this behavior can be changed:
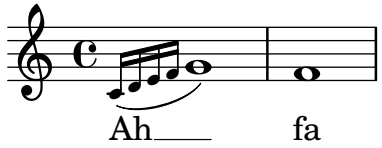
```
<<
  \new Voice = melody \relative c' {
    f4 \appoggiatura a32 b4
    \grace { f16 a16 } b2
    \afterGrace b2 { f16[ a16] }
    \appoggiatura a32 b4
    \acciaccatura a8 b4
  }
  \new Lyrics
  \lyricsto melody {
    normal
    \set includeGraceNotes = ##t
    case,
    gra -- ce case,
    after -- grace case,
    \set ignoreMelismata = ##t
    app. case,
    acc. case.
  }
>>
```



Problemi noti e avvertimenti

Like `associatedVoice`, `includeGraceNotes` needs to be set at latest one syllable before the one which is to be put under a grace note. For the case of a grace note at the very beginning of a piece of music, consider using a `\with` or `\context` block:

```
<<
  \new Voice = melody \relative c' {
    \grace { c16( d e f }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>
```

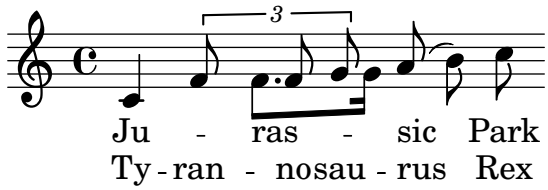


Switching to an alternative melody

More complex variations in setting lyrics to music are possible. The melody to which the lyrics are being set can be changed from within the lyrics by setting the `associatedVoice` property:

```
<<
  \relative c' \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c4
    <<
      \new Voice = "alternative" {
        \voiceOne
        \tuplet 3/2 {
          % show associations clearly.
          \override NoteColumn.force-hshift = #-3
          f8 f g
        }
      }
      {
        \voiceTwo
        f8.[ g16]
        \oneVoice
      } >>
    a8( b) c
  }
  \new Lyrics \lyricsto "lahlah" {
    Ju -- ras -- sic Park
  }
  \new Lyrics \lyricsto "lahlah" {
    % Tricky: need to set associatedVoice
    % one syllable too soon!
    \set associatedVoice = "alternative" % applies to "ran"
    Ty --
```

```
    ran --
    no --
    \set associatedVoice = "lahlah" % applies to "rus"
    sau -- rus Rex
  } >>
```



The text for the first stanza is set to the melody called 'lahlah' in the usual way, but the second stanza is set initally to the `lahlah` context and is then switched to the `alternative` melody for the syllables 'ran' to 'sau' by the lines:

```
\set associatedVoice = "alternative" % applies to "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % applies to "rus"
sau -- rus Rex
```

Here, `alternative` is the name of the `Voice` context containing the triplet.

Note the placement of the `\set associatedVoice` command – it appears to be one syllable too early, but this is correct.

Nota: The `\set associatedVoice` command must be placed one syllable *before* the one at which the switch to the new voice is to occur. In other words, changing the associated Voice happens one syllable later than expected. This is for technical reasons, and it is not a bug.

Printing stanzas at the end

Sometimes it is appropriate to have one stanza set to the music, and the rest added in verse form at the end of the piece. This can be accomplished by adding the extra verses into a `\markup` section outside of the main score block. Notice that there are two different ways to force linebreaks when using `\markup`.

```
melody = \relative c' {
e d c d | e e e e |
d d e d | c1 |
}
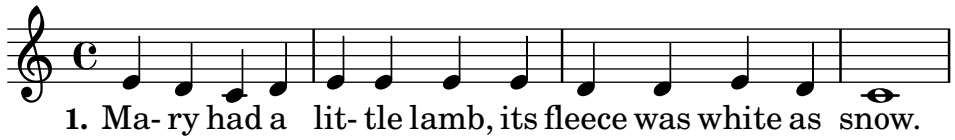
text = \lyricmode {
\set stanza = #"1." Ma- ry had a lit- tle lamb,
its fleece was white as snow.
}

\score{ <<
  \new Voice = "one" { \melody }
  \new Lyrics \lyricsto "one" \text
>>
  \layout { }
}
```

```
\markup { \column{
  \line{ Verse 2. }
  \line{ All the children laughed and played }
  \line{ To see a lamb at school. }
  }
}
\markup{
  \wordwrap-string #"
  Verse 3.

  Mary took it home again,

  It was against the rule."
}
```



Verse 2.
All the children laughed and played
To see a lamb at school.

Verse 3.
Mary took it home again,
It was against the rule.

Printing stanzas at the end in multiple columns

When a piece of music has many verses, they are often printed in multiple columns across the page. An outdented verse number often introduces each verse. The following example shows how to produce such output in LilyPond.
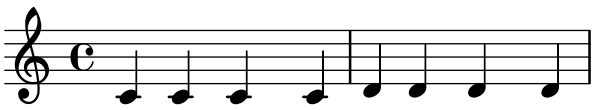
```
melody = \relative c' {
  c4 c c c | d d d d
}

text = \lyricmode {
  \set stanza = #"1." This is verse one.
  It has two lines.
}

\score {
  <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
}

\markup {
```

```
  \fill-line {
    \hspace #0.1 % moves the column off the left margin;
     % can be removed if space on the page is tight
     \column {
      \line { \bold "2."
        \column {
          "This is verse two."
          "It has two lines."
        }
      }
      \combine \null \vspace #0.1 % adds vertical spacing between verses
      \line { \bold "3."
        \column {
          "This is verse three."
          "It has two lines."
        }
      }
    }
    \hspace #0.1 % adds horizontal spacing between columns;
    \column {
      \line { \bold "4."
        \column {
          "This is verse four."
          "It has two lines."
        }
      }
      \combine \null \vspace #0.1 % adds vertical spacing between verses
      \line { \bold "5."
        \column {
          "This is verse five."
          "It has two lines."
        }
      }
    }
  \hspace #0.1 % gives some extra space on the right margin;
  % can be removed if page space is tight
  }
}
```



1. This is verse one. It has two lines.

2. This is verse two.
It has two lines.

3. This is verse three.
It has two lines.

4. This is verse four.
It has two lines.

5. This is verse five.
It has two lines.

Vedi anche

Internals Reference: Sezione “LyricText” in *Guida al Funzionamento Interno*, Sezione “StanzaNumber” in *Guida al Funzionamento Interno*.

2.1.4 Songs

References for songs

Songs are usually written on three staves with the melody for the singer on the top staff and two staves of piano accompaniment at the bottom. The lyrics of the first stanza are printed immediately underneath the top staff. If there are just a small number of further stanzas these can be printed immediately under the first one, but if there are more stanzas than can be easily accommodated there the second and subsequent stanzas are printed after the music as stand-alone text.

All the notational elements needed to write songs are fully described elsewhere:

- For constructing the staff layout, see ⟨undefined⟩ [Displaying staves], pagina ⟨undefined⟩.
- For writing piano music, see Sezione 2.2 [Keyboard and other multi-staff instruments], pagina 310.
- For writing the lyrics to a melody line, see Sezione 2.1.1 [Common notation for vocal music], pagina 247.
- For placing the lyrics, see [Placing lyrics vertically], pagina 261.
- For entering stanzas, see Sezione 2.1.3 [Stanzas], pagina 278.
- Songs are frequently printed with the chording indicated by chord names above the staves. This is described in Sezione 2.7.2 [Displaying chords], pagina 397.
- To print fret diagrams of the chords for guitar accompaniment or accompaniment by other fretted instruments, see “Fret diagram markups” in Sezione 2.4.1 [Common notation for fretted strings], pagina 324.

Vedi anche

Learning Manual: Sezione “Songs” in *Manuale di Apprendimento*.

Notation Reference: Sezione 2.1.1 [Common notation for vocal music], pagina 247, Sezione 2.7.2 [Displaying chords], pagina 397, ⟨undefined⟩ [Displaying staves], pagina ⟨undefined⟩, Sezione 2.2 [Keyboard and other multi-staff instruments], pagina 310, [Placing lyrics vertically], pagina 261, Sezione 2.1.3 [Stanzas], pagina 278.

Snippets: Sezione “Vocal music” in *Frammenti di codice*.

Lead sheets

Lead sheets may be printed by combining vocal parts and ‘chord mode’; this syntax is explained in Sezione 2.7 [Chord notation], pagina 392.

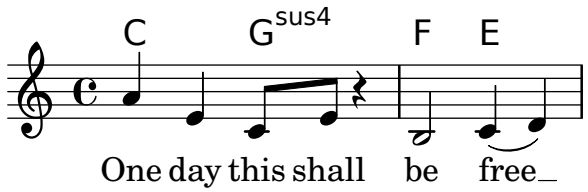
Frammenti di codice selezionati

Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet:

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
```

```
  \addlyrics { One day this shall be free __ }
>>
```



Vedi anche

Notation Reference: Sezione 2.7 [Chord notation], pagina 392.

2.1.5 Choral

This section discusses notation issues that relate most directly to choral music. This includes anthems, part songs, oratorio, etc.

References for choral

Choral music is usually notated on two, three or four staves within a `ChoirStaff` group. Accompaniment, if required, is placed beneath in a `PianoStaff` group, which is usually reduced in size for rehearsal of *a cappella* choral works. The notes for each vocal part are placed in a `Voice` context, with each staff being given either a single vocal part (i.e., one `Voice`) or a pair of vocal parts (i.e., two `Voices`).

Words are placed in `Lyrics` contexts, either underneath each corresponding music staff, or one above and one below the music staff if this contains the music for two parts.

Several common topics in choral music are described fully elsewhere:

- An introduction to creating an SATB vocal score can be found in the Learning Manual, see Sezione “Four-part SATB vocal score” in *Manuale di Apprendimento*.
- Several templates suitable for various styles of choral music can also be found in the Learning Manual, see Sezione “Vocal ensembles templates” in *Manuale di Apprendimento*.
- For information about `ChoirStaff` and `PianoStaff` see ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩.
- Shape note heads, as used in Sacred Harp and similar notation, are described in ⟨undefined⟩ [Shape note heads], pagina ⟨undefined⟩.
- When two vocal parts share a staff the stems, ties, slurs, etc., of the higher part will be directed up and those of the lower part down. To do this, use `\voiceOne` and `\voiceTwo`. See ⟨undefined⟩ [Single-staff polyphony], pagina ⟨undefined⟩.
- When a vocal part temporarily splits, you should use *Temporary polyphonic passages* (see ⟨undefined⟩ [Single-staff polyphony], pagina ⟨undefined⟩).

Comandi predefiniti

`\oneVoice`, `\voiceOne`, `\voiceTwo`.

Vedi anche

Learning Manual: Sezione “Four-part SATB vocal score” in *Manuale di Apprendimento*, Sezione “Vocal ensembles templates” in *Manuale di Apprendimento*.

Notation Reference: Sezione 5.1.7 [Context layout order], pagina 570, ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩, ⟨undefined⟩ [Shape note heads], pagina ⟨undefined⟩, ⟨undefined⟩ [Single-staff polyphony], pagina ⟨undefined⟩.

Snippets: Sezione “Vocal music” in *Frammenti di codice*.

Internals Reference: Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

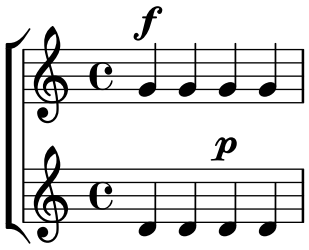
Score layouts for choral

Choral music containing four staves, with or without piano accompaniment, is usually laid out with two systems per page. Depending on the page size, achieving this may require changes to several default settings. The following settings should be considered:

- The global staff size can be modified to change the overall size of the elements of the score. See Sezione 4.2.2 [Setting the staff size], pagina 514.
- The distances between the systems, the staves and the lyrics can all be adjusted independently. See Sezione 4.4 [Vertical spacing], pagina 523.
- The dimensions of the vertical layout variables can be displayed as an aid to adjusting the vertical spacing. This and other possibilities for fitting the music onto fewer pages are described in Sezione 4.6 [Fitting music onto fewer pages], pagina 549.
- If the number of systems per page changes from one to two it is customary to indicate this with a system separator mark between the two systems. See ⟨undefined⟩ [Separating systems], pagina ⟨undefined⟩.
- For details of other page formatting properties, see Sezione 4.1 [Page layout], pagina 502.

Dynamic markings by default are placed below the staff, but in choral music they are usually placed above the staff in order to avoid the lyrics. The predefined command `\dynamicUp` does this for the dynamic markings in a single `Voice` context. If there are many `Voice` contexts this predefined command would have to be placed in every one. Alternatively its expanded form can be used to place all dynamic markings in the entire score above their respective staves, as shown here:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice {
        \relative c'' { g4\f g g g }
      }
    }
    \new Staff {
      \new Voice {
        \relative c' { d4 d d\p d }
      }
    }
  >>
  \layout {
    \context {
      \Score
      \override DynamicText.direction = #UP
      \override DynamicLineSpanner.direction = #UP
    }
  }
}
```



Comandi predefiniti

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

Vedi anche

Notation Reference: Sezione 4.6.2 [Changing spacing], pagina 550, Sezione 4.6.1 [Displaying spacing], pagina 549, Sezione 4.6 [Fitting music onto fewer pages], pagina 549, Sezione 4.1 [Page layout], pagina 502, Sezione 4.2 [Score layout], pagina 512, ⟨undefined⟩ [Separating systems], pagina ⟨undefined⟩, Sezione 4.2.2 [Setting the staff size], pagina 514, Sezione 4.3.8 [Using an extra voice for breaks], pagina 521, Sezione 4.4 [Vertical spacing], pagina 523.

Internals Reference: Sezione "VerticalAxisGroup" in *Guida al Funzionamento Interno*, Sezione "StaffGrouper" in *Guida al Funzionamento Interno*.

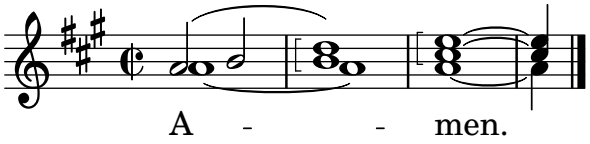
Divided voices

Using arpeggioBracket to make divisi more visible

The `arpeggioBracket` can be used to indicate the division of voices where there are no stems to provide the information. This is often seen in choral music.

```
\include "english.ly"

\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
          <b d>1\arpeggio)
          <cs e>\arpeggio ~
          <cs e>4
        }
        \addlyrics { \lyricmode { A -- men. } }
      >>
      \new Voice = "lower"
      { \voiceTwo
        a1 ~
        a
        a ~
        a4 \bar "|."
      }
    >>
  }
  \layout { ragged-right = ##t }
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Expressive marks as lines], pagina ⟨undefined⟩.

2.1.6 Opera and stage musicals

The music, lyrics and dialogue to opera and stage musicals are usually set out in one or more of the following forms:

- A *Conductors' Score* containing the full orchestral and vocal parts, together with libretto cues if there are spoken passages.
- *Orchestral Parts* containing the music for the individual instruments of the orchestra or band.
- A *Vocal Score* containing all vocal parts with piano accompaniment. The accompaniment is usually an orchestral reduction, and if so the name of the original orchestral instrument is often indicated. Vocal scores sometimes includes stage directions and libretto cues.
- A *Vocal Book* containing just the vocal parts (no accompaniment), sometimes combined with the libretto.
- A *Libretto* containing the extended passages of spoken dialogue usually found in musicals, together with the words to the sung parts. Stage directions are usually included. LilyPond can be used to typeset libretti but as they contain no music alternative methods may be preferable.

The sections in the LilyPond documentation which cover the topics needed to create scores in the styles commonly found in opera and musicals are indicated in the References below. This is followed by sections covering those techniques which are peculiar to typesetting opera and musical scores.

References for opera and stage musicals

- A conductors' score contains many grouped staves and lyrics. Ways of grouping staves is shown in ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩. To nest groups of staves see ⟨undefined⟩ [Nested staff groups], pagina ⟨undefined⟩.
- The printing of empty staves in conductors' scores and vocal scores is often suppressed. To create such a "Frenched score" see ⟨undefined⟩ [Hiding staves], pagina ⟨undefined⟩.
- Writing orchestral parts is covered in ⟨undefined⟩ [Writing parts], pagina ⟨undefined⟩. Other sections in the Specialist notation chapter may be relevant, depending on the orchestration used. Many instruments are transposing instruments, see ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩.
- If the number of systems per page changes from page to page it is customary to separate the systems with a system separator mark. See ⟨undefined⟩ [Separating systems], pagina ⟨undefined⟩.
- For details of other page formatting properties, see Sezione 4.1 [Page layout], pagina 502.
- Dialogue cues, stage directions and footnotes can be inserted, see Sezione 3.2.3 [Creating footnotes], pagina 468 and ⟨undefined⟩ [Text], pagina ⟨undefined⟩. Extensive stage directions can also be added with a section of stand-alone markups between two `\score` blocks, see ⟨undefined⟩ [Separate text], pagina ⟨undefined⟩.

Vedi anche

Musical Glossary: Sezione "Frenched score" in *Glossario Musicale*, Sezione "Frenched staves" in *Glossario Musicale*, Sezione "transposing instrument" in *Glossario Musicale*.

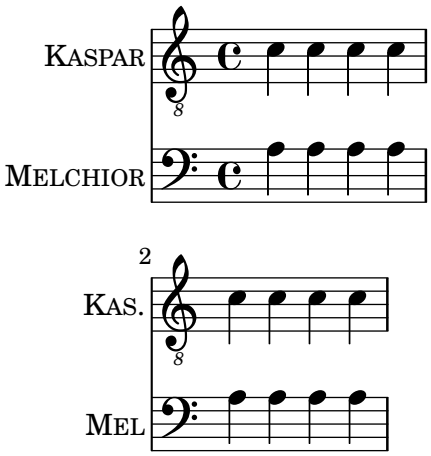
Notation Reference: Sezione 3.2.3 [Creating footnotes], pagina 468, ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩, ⟨undefined⟩ [Hiding staves], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩, ⟨undefined⟩ [Nested staff groups], pagina ⟨undefined⟩, Sezione 4.1 [Page layout], pagina 502, ⟨undefined⟩ [Separating systems], pagina ⟨undefined⟩, ⟨undefined⟩ [Transpose], pagina ⟨undefined⟩, ⟨undefined⟩ [Writing parts], pagina ⟨undefined⟩, ⟨undefined⟩ [Writing text], pagina ⟨undefined⟩.

Snippets: Sezione "Vocal music" in *Frammenti di codice*.

Character names
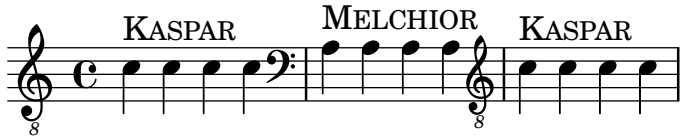
Character names are usually shown to the left of the staff when the staff is dedicated to that character alone:

```
\score {
  <<
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Kaspar
      \set Staff.shortVocalName = \markup \smallCaps Kas.
      \relative c' {
        \clef "G_8"
        c4 c c c
        \break
        c4 c c c
      }
    }
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Melchior
      \set Staff.shortVocalName = \markup \smallCaps Mel
      \clef "bass"
      \relative c' {
        a4 a a a
        a4 a a a
      }
    }
  >>
}
```



When two or more characters share a staff the character's name is usually printed above the staff at the start of every section applying to that character. This can be done with markup. Often a specific font is used for this purpose.

```
\clef "G_8"
c4^\markup \fontsize #1 \smallCaps Kaspar
c c c
\clef "bass"
a4^\markup \fontsize #1 \smallCaps Melchior
a a a
\clef "G_8"
c4^\markup \fontsize #1 \smallCaps Kaspar
c c c
```
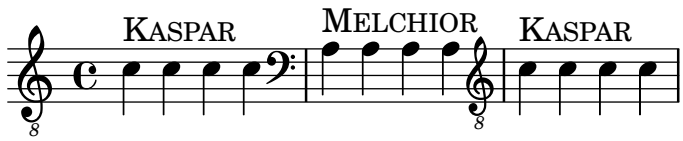


Alternatively, if there are many character changes, it may be easier to set up "instrument" definitions for each character at the top level so that `\instrumentSwitch` can be used to indicate each change.

```
\addInstrumentDefinition #"kaspar"
  #`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
     (shortInstrumentName . "Kas.")
     (clefGlyph . "clefs.G")
     (clefTransposition . -7)
     (middleCPosition . 1)
     (clefPosition . -2)
     (instrumentCueName . ,(markup #:fontsize 1 #:smallCaps "Kaspar"))
     (midiInstrument . "voice oohs"))

\addInstrumentDefinition #"melchior"
  #`((instrumentTransposition . ,(ly:make-pitch 0 0 0))
     (shortInstrumentName . "Mel.")
     (clefGlyph . "clefs.F")
     (clefTransposition . 0)
     (middleCPosition . 6)
     (clefPosition . 2)
     (instrumentCueName . ,(markup #:fontsize 1 #:smallCaps "Melchior"))
     (midiInstrument . "choir aahs"))

\relative c' {
  \instrumentSwitch "kaspar"
  c4 c c c
  \instrumentSwitch "melchior"
  a4 a a a
  \instrumentSwitch "kaspar"
  c4 c c c
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩, Sezione A.21 [Scheme functions], pagina 754, ⟨undefined⟩ [Text], pagina ⟨undefined⟩, Sezione A.10 [Text markup commands], pagina 653.

Extending LilyPond: Sezione “Markup construction in Scheme” in *Estendere*.

Musical cues

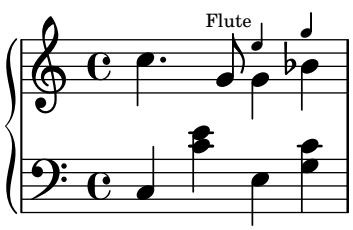
Musical cues can be inserted in Vocal Scores, Vocal Books and Orchestral Parts to indicate what music in another part immediately precedes an entry. Also, cues are often inserted in the piano reduction in Vocal Scores to indicate what each orchestral instrument is playing. This aids the conductor when a full Conductors' Score is not available.

The basic mechanism for inserting cues is fully explained in the main text, see ⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩ and ⟨undefined⟩ [Formatting cue notes], pagina ⟨undefined⟩. But when many cues have to be inserted, for example, as an aid to a conductor in a vocal score, the instrument name must be positioned carefully just before and close to the start of the cue notes. The following example shows how this is done.

```
flute = \relative c'' {
  s4 s4 e g
}
\addQuote "flute" { \flute }

pianoRH = \relative c'' {
  c4. g8
  % position name of cue-ing instrument just before the cue notes,
  % and above the staff
  \new CueVoice {
    \override InstrumentSwitch.self-alignment-X = #RIGHT
    \set instrumentCueName = "Flute"
  }
  \cueDuring "flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  \new PianoStaff <<
    \new Staff {
      \pianoRH
    }
    \new Staff {
      \clef "bass"
      \pianoLH
    }
  >>
}
```

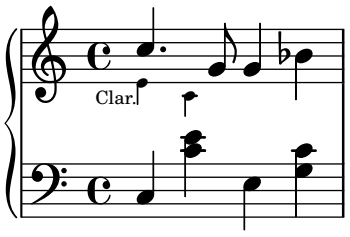


If a transposing instrument is being quoted the instrument part should specify its key so the conversion of its cue notes will be done automatically. The example below shows this transposition for a B-flat clarinet. The notes in this example are low on the staff so `DOWN` is specified in `\cueDuring` (so the stems are down) and the instrument name is positioned below the staff. Note also that the piano right-hand voice is explicitly declared. This is because the cue notes in this example begin at the start of the first bar and this would otherwise cause the entire piano right-hand notes to be placed in a `CueVoice` context.

```
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % position name of cue-ing instrument below the staff
  \new CueVoice {
    \override InstrumentSwitch.self-alignment-X = #RIGHT
    \override InstrumentSwitch.direction = #DOWN
    \set instrumentCueName = "Clar."
  }
  \cueDuring "clarinet" #DOWN { c4. g8 }
  g4 bes4
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



From these two examples it is clear that inserting many cues in a Vocal Score would be tedious, and the notes of the piano part would become obscured. However, as the following

snippet shows, it is possible to define a music function to reduce the amount of typing and to make the piano notes clearer.

Frammenti di codice selezionati

Adding orchestral cues to a vocal score

This shows one approach to simplify adding many orchestral cues to the piano reduction in a vocal score. The music function `\cueWhile` takes four arguments: the music from which the cue is to be taken, as defined by `\addQuote`, the name to be inserted before the cue notes, then either `#UP` or `#DOWN` to specify either `\voiceOne` with the name above the staff or `\voiceTwo` with the name below the staff, and finally the piano music in parallel with which the cue notes are to appear. The name of the cued instrument is positioned to the left of the cued notes. Many passages can be cued, but they cannot overlap each other in time.

```
cueWhile =
#(define-music-function
   (parser location instrument name dir music)
   (string? string? ly:dir? ly:music?)
   #{
     \cueDuring $instrument #dir {
       \once \override TextScript.self-alignment-X = #RIGHT
       \once \override TextScript.direction = $dir
       <>-\markup { \tiny #name }
       $music
     }
   #})

flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
\addQuote "flute" { \flute }
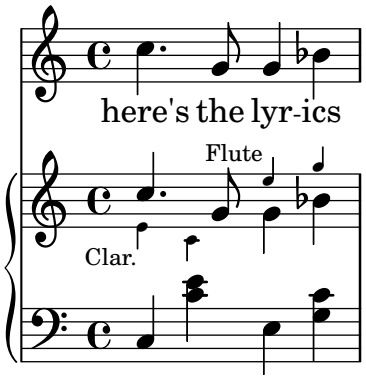
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
```

```
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



Vedi anche

Musical Glossary: Sezione “cue-notes” in *Glossario Musicale*.

Notation Reference: Sezione 5.5.1 [Aligning objects], pagina 600, Sezione 5.4.2 [Direction and placement], pagina 585, ⟨undefined⟩ [Formatting cue notes], pagina ⟨undefined⟩, ⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩, Sezione 5.6 [Using music functions], pagina 612.

Snippets: Sezione “Vocal music” in *Frammenti di codice*.

Internals Reference: Sezione “InstrumentSwitch” in *Guida al Funzionamento Interno*, Sezione “CueVoice” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

`\cueDuring` automatically inserts a `CueVoice` context and all cue notes are placed in that context. This means it is not possible to have two overlapping sequences of cue notes by this technique. Overlapping sequences could be entered by explicitly declaring separate `CueVoice` contexts and using `\quoteDuring` to extract and insert the cue notes.

Spoken music
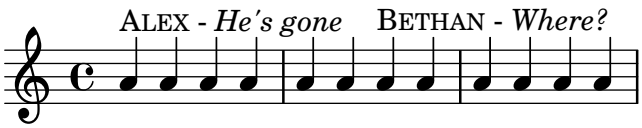
Such effects as ‘parlato’ or ‘Sprechgesang’ require performers to speak without pitch but still with rhythm; these are notated by cross note heads, as demonstrated in ⟨undefined⟩ [Special note heads], pagina ⟨undefined⟩.

Dialogue over music

Dialogue over music is usually printed over the staves in an italic font, with the start of each phrase keyed in to a particular music moment.

For short interjections a simple markup suffices.

```
a4^\markup { \smallCaps { Alex - } \italic { He's gone } } a a a
a4 a a^\markup { \smallCaps { Bethan - } \italic Where? } a
a4 a a a
```



For longer phrases it may be necessary to expand the music to make the words fit neatly. There is no provision in LilyPond to do this fully automatically, and some manual intervention to layout the page will be necessary.

For long phrases or for passages with a lot of closely packed dialogue, using a Lyrics context will give better results. The Lyrics context should not be associated with a music Voice; instead each section of dialogue should be given an explicit duration. If there is a gap in the dialogue, the final word should be separated from the rest and the duration split between them so that the underlying music spaces out smoothly.

If the dialogue extends for more than one line it will be necessary to manually insert `\break`s and adjust the placing of the dialogue to avoid running into the right margin. The final word of the last measure on a line should also be separated out, as above.
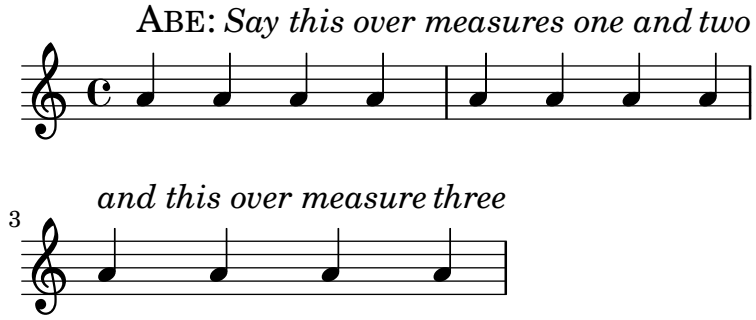
Here is an example illustrating how this might be done.

```
music = \relative c'' {
  \repeat unfold 3 { a4 a a a }
}

dialogue = \lyricmode {
  \markup {
    \fontsize #1 \upright \smallCaps Abe:
    "Say this over measures one and"
  }4*7
  "two"4 |
  \break
  "and this over measure"4*3
  "three"4 |
}

\score {
  <<
    \new Lyrics \with {
      \override LyricText.font-shape = #'italic
      \override LyricText.self-alignment-X = #LEFT
    }
```

```
    { \dialogue }
    \new Staff {
      \new Voice { \music }
    }
  >>
}
```



Vedi anche

Notation Reference: [Manual syllable durations], pagina 253, ⟨undefined⟩ [Text], pagina ⟨undefined⟩.

Internal Reference: Sezione “LyricText” in *Guida al Funzionamento Interno*.

2.1.7 Chants psalms and hymns

The music and words for chants, psalms and hymns usually follow a well-established format in any particular church. Although the formats may differ from church to church the type-setting problems which arise are broadly similar, and are covered in this section.

References for chants and psalms

Typesetting Gregorian chant in various styles of ancient notation is described in Sezione 2.9 [Ancient notation], pagina 413.

Vedi anche

Notation reference: Sezione 2.9 [Ancient notation], pagina 413.

Snippets: Sezione “Vocal music” in *Frammenti di codice*.

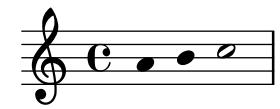
Setting a chant

Modern chant settings use modern notation with varying numbers of elements taken from ancient notation. Some of the elements and methods to consider are shown here.

Chants often use quarter notes without stems to indicate the pitch, with the rhythm being taken from the spoken rhythm of the words.

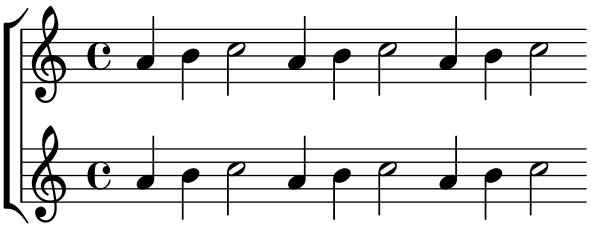
```
stemOff = { \hide Staff.Stem }

\relative c' {
  \stemOff
  a'4 b c2 |
}
```



Chants often omit the bar lines or use shortened or dotted bar lines to indicate pauses in the music. To omit all bar lines from all staves remove the bar line engraver completely:
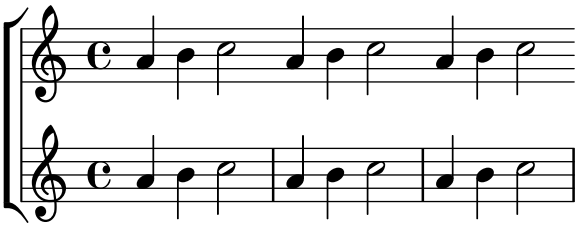
```
\score {
  \new StaffGroup <<
    \new Staff {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}
```



Bar lines can also be removed on a staff-by-staff basis:
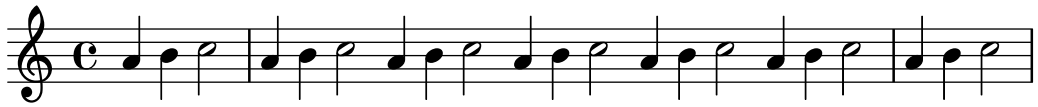
```
\score {
  \new ChoirStaff <<
    \new Staff
    \with { \remove "Bar_engraver" } {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
```
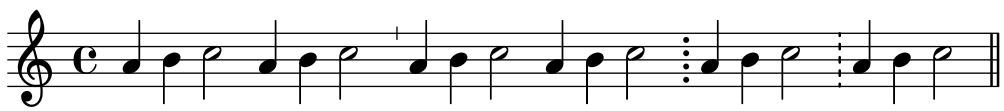
```
    }
  >>
}
```



To remove bar lines from just a section of music treat it as a cadenza. If the section is long you may need to insert dummy bar lines with `\bar ""` to show where the line should break.

```
a4 b c2 |
\cadenzaOn
a4 b c2
a4 b c2
\bar ""
a4 b c2
a4 b c2
\cadenzaOff
a4 b c2 |
a4 b c2 |
```



Rests or pauses in chants can be indicated by modified bar lines.

```
a4
\cadenzaOn
b c2
a4 b c2
\bar "'"
a4 b c2
a4 b c2
\bar ";"
a4 b c2
\bar "!"
a4 b c2
\bar "||"
```
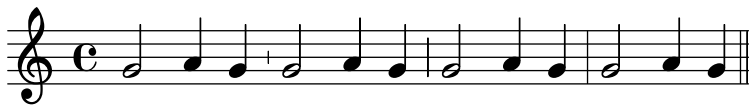


Alternatively, the notation used in Gregorian chant for pauses or rests is sometimes used even though the rest of the notation is modern. This uses a modified `\breathe` mark:

```
divisioMinima = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-minima
  \once \override BreathingSign.Y-offset = #0
```

```
  \breathe
}
divisioMaior = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maior
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
divisioMaxima = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maxima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
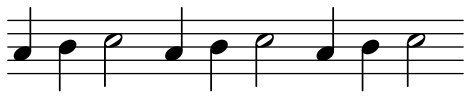
\score {
  \relative c'' {
    g2 a4 g
    \divisioMinima
    g2 a4 g
    \divisioMaior
    g2 a4 g
    \divisioMaxima
    g2 a4 g
    \finalis
  }
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}
```



Chants usually omit the time signature and often omit the clef too.

```
\score {
  \new Staff {
    \relative c'' {
      a4 b c2 |
      a4 b c2 |
      a4 b c2 |
    }
  }
  \layout {
    \context {
```

```
      \Staff
      \remove "Bar_engraver"
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
    }
  }
}
```



Chants for psalms in the Anglican tradition are usually either *single*, with 7 bars of music, or *double*, with two lots of 7 bars. Each group of 7 bars is divided into two halves, corresponding to the two halves of each verse, usually separated by a double bar line. Only whole and half notes are used. The 1st bar in each half always contains a single chord of whole notes. This is the "reciting note". Chants are usually centered on the page.

```
SopranoMusic = \relative g' {
  g1 | c2 b | a1 | \bar "||"
  a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative c' {
  e1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

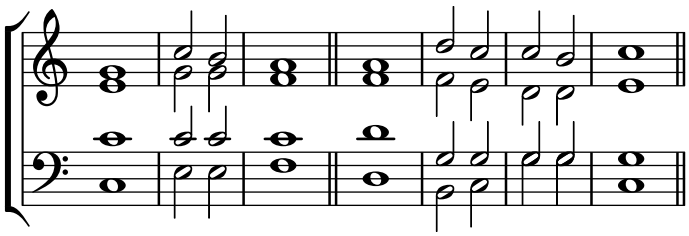
TenorMusic = \relative a {
  c1 | c2 c | c1 |
  d1 | g,2 g | g g | g1 |
}

BassMusic =  \relative c {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
}

% Use markup to center the chant on the page
\markup {
  \fill-line {
    \score {  % centered
      <<
        \new ChoirStaff <<
          \new Staff <<
            \global
            \clef "treble"
            \new Voice = "Soprano" <<
              \voiceOne
```

```
            \SopranoMusic
          >>
          \new Voice = "Alto" <<
            \voiceTwo
            \AltoMusic
          >>
        >>
        \new Staff <<
          \clef "bass"
          \global
          \new Voice = "Tenor" <<
            \voiceOne
            \TenorMusic
          >>
          \new Voice = "Bass" <<
            \voiceTwo
            \BassMusic
          >>
        >>
      >>
      \layout {
        \context {
          \Score
          \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/2)
        }
        \context {
          \Staff
          \remove "Time_signature_engraver"
        }
      }
    }  % End score
  }
}  % End markup
```



Some other approaches to setting such a chant are shown in the first of the following snippets.

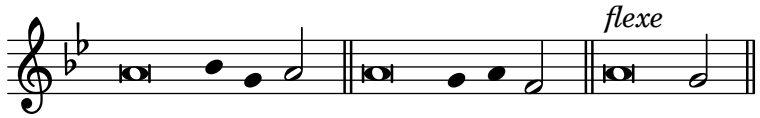
Frammenti di codice selezionati

Chant or psalms notation

This form of notation is used for the chant of the Psalms, where verses aren't always the same length.

```
stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff
```

```
\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve^\markup { \italic flexe }
    \stemOn g'2 \bar "||"
  }
}
```



Canticles and other liturgical texts may be set more freely, and may use notational elements from ancient music. Often the words are shown underneath and aligned with the notes. If so, the notes are spaced in accordance with the syllables rather than the notes' durations.

Modello per notazione antica – trascrizione moderna di musica gregoriana

Questo esempio mostra come realizzare una trascrizione moderna di musica gregoriana. La musica gregoriana non presenta la suddivisione in misure né gambi; impiega soltanto le teste della minima e della semiminima, e dei segni appositi che indicano pause di diversa lunghezza.
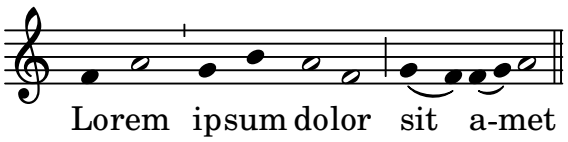
```
\include "gregorian.ly"

chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
```

```
      \Voice
      \override Stem.length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```



Vedi anche

Learning Manual: Sezione “Visibility and color of objects” in *Manuale di Apprendimento*, Sezione “Vocal ensembles templates” in *Manuale di Apprendimento*.

Notation Reference: Sezione 2.9 [Ancient notation], pagina 413, ⟨undefined⟩ [Bar lines], pagina ⟨undefined⟩, Sezione 5.1.4 [Modifying context plug-ins], pagina 561, Sezione 2.9.4 [Typesetting Gregorian chant], pagina 424, ⟨undefined⟩ [Unmetered music], pagina ⟨undefined⟩, Sezione 5.4.6 [Visibility of objects], pagina 592.

Pointing a psalm

The words to an Anglican psalm are usually printed in separate verses centered underneath the chant.

Single chants (with 7 bars) are repeated for every verse. Double chants (with 14 bars) are repeated for every pair of verses. Marks are inserted in the words to show how they should be fitted to the chant. Each verse is divided into two halves. A colon is usually used to indicate this division. This corresponds to the double bar line in the music. The words before the colon are sung to the first three bars of music; the words after the colon are sung to the last four bars.

Single bar lines (or in some psalters an inverted comma or similar symbol) are inserted between words to indicate where the bar lines in the music fall. In markup mode a single bar line can be entered with the bar check symbol, `|`.

```
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing | unto the | Lord : let }
        \line { us heartily rejoice in the | strength of | our }
        \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

Other symbols may require glyphs from the `fetaMusic` fonts. For details, see ⟨undefined⟩ [Fonts], pagina ⟨undefined⟩.

```
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing \tick unto the \tick Lord : let }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```

O come let us sing ' unto the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

Where there is one whole note in a bar all the words corresponding to that bar are recited on that one note in speech rhythm. Where there are two notes in a bar there will usually be only one or two corresponding syllables. If there are more that two syllables a dot is usually inserted to indicate where the change in note occurs.

```
dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us sing \tick unto \dot the \tick Lord : let
        }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```

O come let us sing ' unto • the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

In some psalters an asterisk is used to indicate a break in a recited section instead of a comma, and stressed or slightly lengthened syllables are indicated in bold text.

```
dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { Today if ye will hear his voice * }
        \line {
          \concat { \bold hard en }
          | not your | hearts : as in the pro-
        }
        \line { vocation * and as in the \bold day of tempt- | }
        \line { -ation | in the | wilderness. }
      }
    }
  }
}
```

Today if ye will hear his voice *
harden | not your | hearts : as in the pro-
vocation * and as in the **day** of tempt- |
-ation | in the | wilderness.

In other psalters an accent is placed over the syllable to indicate stress.

```
tick = \markup {
  \raise #2 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us \concat {
            si \combine \tick ng
          }
          | unto the | Lord : let
        }
        \line {
          us heartily \concat {
            rejo \combine \tick ice
          }
          in the | strength of | our
        }
        \line { sal- | -vation. }
      }
```

```
    }
  }
}
```

O come let us sing′ | unto the | Lord : let
us heartily rejoice′ in the | strength of | our
sal- | -vation.

The use of markup to center text, and arrange lines in columns is described in ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Most of these elements are shown in one or other of the two verses in the template, see Sezione "Psalms" in *Manuale di Apprendimento*.

Vedi anche

Learning Manual: Sezione "Psalms" in *Manuale di Apprendimento*, Sezione "Vocal ensembles templates" in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Fonts], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Partial measures in hymn tunes

Hymn tunes frequently start and end every line of music with partial measures so that each line of music corresponds exactly with a line of text. This requires a `\partial` command at the start of the music and `\bar "|"` or `\bar "||"` commands at the end of each line.

Modello per inno

Il codice seguente presenta un modo di impostare un inno in cui ogni verso inizia e finisce con una misura parziale. Mostra anche come aggiungere delle strofe come testo separato sotto la musica.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}
```

```
BassMusic =  \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

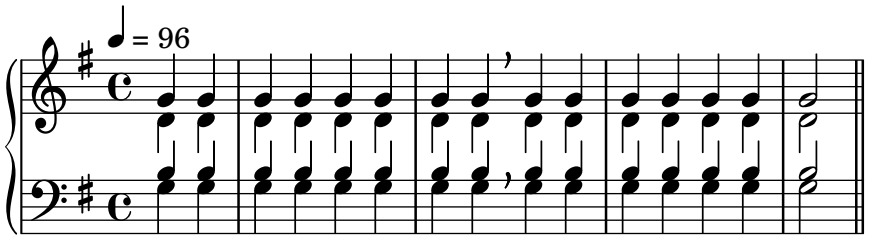
global = {
 \key g \major
}

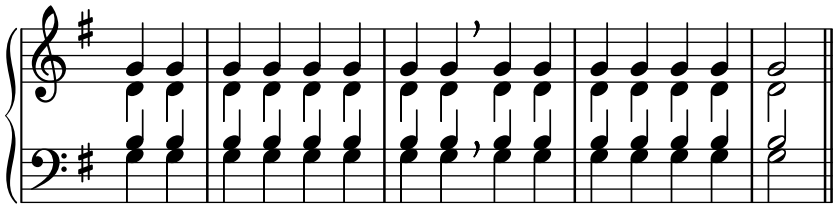
\score {  % Start score
  <<
    \new PianoStaff <<  % Start pianostaff
      \new Staff <<  % Start Staff = RH
        \global
        \clef "treble"
        \new Voice = "Soprano" <<  % Start Voice = "Soprano"
          \Timeline
          \voiceOne
          \SopranoMusic
        >>  % End Voice = "Soprano"
        \new Voice = "Alto" <<  % Start Voice = "Alto"
          \Timeline
          \voiceTwo
          \AltoMusic
        >>  % End Voice = "Alto"
      >>  % End Staff = RH
      \new Staff <<  % Start Staff = LH
        \global
        \clef "bass"
        \new Voice = "Tenor" <<  % Start Voice = "Tenor"
          \Timeline
          \voiceOne
          \TenorMusic
        >>  % End Voice = "Tenor"
        \new Voice = "Bass" <<  % Start Voice = "Bass"
          \Timeline
          \voiceTwo
          \BassMusic
        >>  % End Voice = "Bass"
      >>  % End Staff = LH
    >>  % End pianostaff
  >>
}  % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
```

```
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
    ""
  }
}

\paper {  % Start paper block
  indent = 0     % don't indent first system
  line-width = 130   % shorten line length to suit music
}  % End paper block
```





This is line one of the first verse
This is line two of the same
And here's line three of the first verse
And the last line of the same

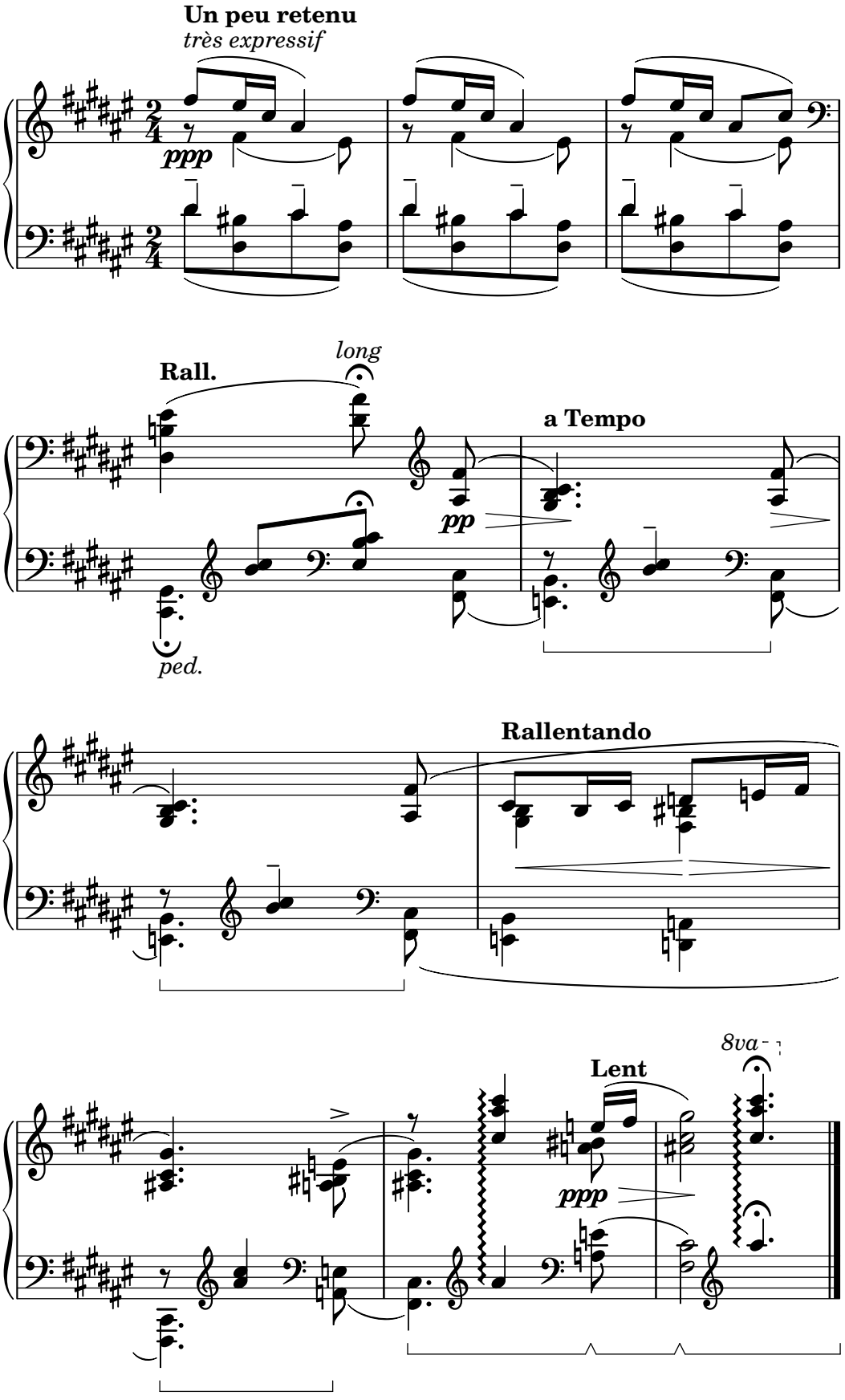
2.1.8 Ancient vocal music

Ancient vocal music is supported, as explained in Sezione 2.9 [Ancient notation], pagina 413.

Vedi anche

Notation Reference: Sezione 2.9 [Ancient notation], pagina 413.

2.2 Keyboard and other multi-staff instruments



This section discusses several aspects of music notation that are unique to keyboard instruments and other instruments notated on many staves, such as harps and vibraphones. For the purposes of this section this entire group of multi-staff instruments is called "keyboards" for short, even though some of them do not have a keyboard.

2.2.1 Common notation for keyboards

This section discusses notation issues that may arise for most keyboard instruments.

References for keyboards

Keyboard instruments are usually notated with Piano staves. These are two or more normal staves coupled with a brace. The same notation is also used for other keyed instruments. Organ music is normally written with two staves inside a `PianoStaff` group and third, normal staff for the pedals.

The staves in keyboard music are largely independent, but sometimes voices can cross between the two staves. This section discusses notation techniques particular to keyboard music.

Several common issues in keyboard music are covered elsewhere:

- Keyboard music usually contains multiple voices and the number of voices may change regularly; this is described in ⟨undefined⟩ [Collision resolution], pagina ⟨undefined⟩.
- Keyboard music can be written in parallel, as described in ⟨undefined⟩ [Writing music in parallel], pagina ⟨undefined⟩.
- Dynamics may be placed in a `Dynamics` context, between the two `Staff` contexts to align the dynamic marks on a horizontal line centered between the staves; see ⟨undefined⟩ [Dynamics], pagina ⟨undefined⟩.
- Fingerings are indicated with ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩.
- Organ pedal indications are inserted as articulations, see Sezione A.13 [List of articulations], pagina 706.
- Vertical grid lines can be shown with ⟨undefined⟩ [Grid lines], pagina ⟨undefined⟩.
- Keyboard music often contains *Laissez vibrer* ties as well as ties on arpeggios and tremolos, described in ⟨undefined⟩ [Ties], pagina ⟨undefined⟩.
- Placing arpeggios across multiple voices and staves is covered in [Arpeggio], pagina 137.
- Tremolo marks are described in ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩.
- Several of the tweaks that can occur in keyboard music are demonstrated in Sezione "Real music example" in *Manuale di Apprendimento*.
- Hidden notes can be used to produce ties that cross voices, as shown in Sezione "Other uses for tweaks" in *Manuale di Apprendimento*.

Vedi anche

Learning Manual: Sezione "Real music example" in *Manuale di Apprendimento*, Sezione "Other uses for tweaks" in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩, ⟨undefined⟩ [Collision resolution], pagina ⟨undefined⟩, ⟨undefined⟩ [Writing music in parallel], pagina ⟨undefined⟩, ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩, Sezione A.13 [List of articulations], pagina 706, ⟨undefined⟩ [Grid lines], pagina ⟨undefined⟩, ⟨undefined⟩ [Ties], pagina ⟨undefined⟩, [Arpeggio], pagina 137, ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩.

Internals Reference: Sezione "PianoStaff" in *Guida al Funzionamento Interno*.

Snippets: Sezione "Keyboards" in *Frammenti di codice*.

Changing staff manually

Voices can be switched between staves manually, using the command
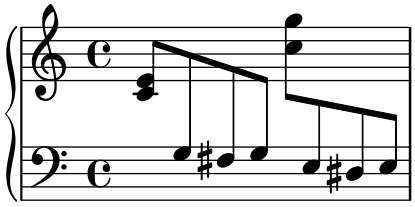
`\change Staff = staffname`

The string *staffname* is the name of the staff. It switches the current voice from its current staff to the staff called *staffname*. Typical values for *staffname* are `"up"` and `"down"`, or `"RH"` and `"LH"`.

The staff to which the voice is being switched must exist at the time of the switch. If necessary, staves should be "kept alive", see Sezione 5.1.3 [Keeping contexts alive], pagina 558.

Cross-staff notes are beamed automatically:

```
\new PianoStaff <<
  \new Staff = "up" {
    <e' c'>8
    \change Staff = "down"
    g8 fis g
    \change Staff = "up"
    <g'' c''>8
    \change Staff = "down"
    e8 dis e
    \change Staff = "up"
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```

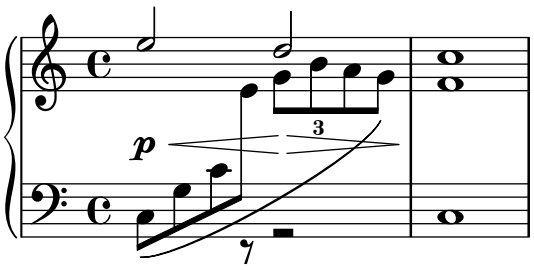


If the beaming needs to be tweaked, make any changes to the stem directions first. The beam positions are then measured from the center of the staff that is closest to the beam. For a simple example of beam tweaking, see Sezione "Fixing overlapping notation" in *Manuale di Apprendimento*.

Overlapping notation can result when voices cross staves:

```
\new PianoStaff <<
  \new Staff = "up" {
    \voiceOne
    % Make space for fingering in the cross-staff voice
    \once\override DynamicLineSpanner.staff-padding = #4
    e''2\p\< d''\>
    c''1\!
  }
  \new Staff = "down" <<
    {
      \clef bass
```

```
      s4. e,8\rest g,2\rest
      c1
    } \\ {
      c8\( g c'
      \change Staff = "up"
      e' g' b'-3 a' g'\)
      f'1
    }
  >>
>>
```



The stem and slur overlap the intervening line of dynamics because automatic collision resolution is suspended for beams, slurs and other spanners that connect notes on different staves, as well as for stems and articulations if their placement is affected by a cross-staff spanner. The resulting collisions must be resolved manually, where necessary, using the methods in Sezione "Fixing overlapping notation" in *Manuale di Apprendimento*.

Vedi anche

Learning Manual: Sezione "Fixing overlapping notation" in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Stems], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic beams], pagina ⟨undefined⟩, Sezione 5.1.3 [Keeping contexts alive], pagina 558.

Snippets: Sezione "Keyboards" in *Frammenti di codice*.

Internals Reference: Sezione "Beam" in *Guida al Funzionamento Interno*, Sezione "ContextChange" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Beam collision avoidance does not work for automatic beams that end right before a change in staff. In this case use manual beams.

Changing staff automatically
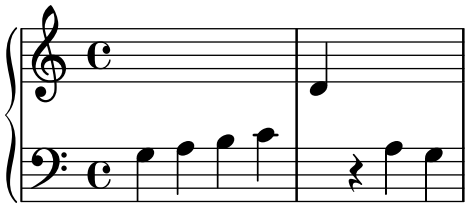
Voices can be made to switch automatically between the top and the bottom staff. The syntax for this is

`\autochange ...`*`music`*`...`

This will create two staves inside the current staff group (usually a `PianoStaff`), called `"up"` and `"down"`. The lower staff will be in the bass clef by default. The autochanger switches on the basis of the pitch (middle C is the turning point), and it looks ahead skipping over rests to switch in advance.

```
\new PianoStaff {
  \autochange {
    g4 a b c'
    d'4 r a g
  }
```

```
}
```



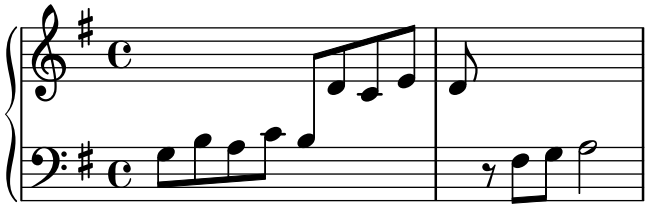
A `\relative` section that is outside of `\autochange` has no effect on the pitches of the music, so if necessary, put `\relative` inside `\autochange`.

If additional control is needed over the individual staves, they can be created manually with the names `"up"` and `"down"`. The `\autochange` command will then switch its voice between the existing staves.

Nota: If staves are created manually, they *must* be named `"up"` and `"down"`.

For example, staves must be created manually in order to place a key signature in the lower staff:

```
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melOne" {
      \key g \major
      \autochange \relative c' {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



Vedi anche

Notation Reference: [Changing staff manually], pagina 312.

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “AutoChangeMusic” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

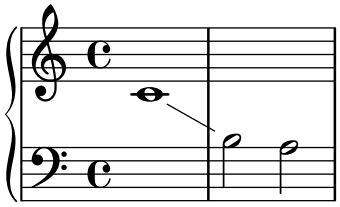
The staff switches may not end up in optimal places. For high quality output, staff switches should be specified manually.

Chords will not be split across the staves; they will be assigned to a staff based on the first note named in the chord construct.

Staff-change lines

Whenever a voice switches to another staff, a line connecting the notes can be printed automatically:

```
\new PianoStaff <<
  \new Staff = "one" {
    \showStaffSwitch
    c1
    \change Staff = "two"
    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>
```



Comandi predefiniti

`\showStaffSwitch`, `\hideStaffSwitch`.

Vedi anche

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “Note_head_line_engraver” in *Guida al Funzionamento Interno*, Sezione “VoiceFollower” in *Guida al Funzionamento Interno*.

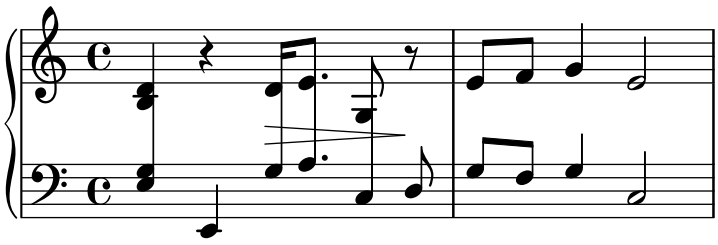
Cross-staff stems

Chords that cross staves may be produced using the `Span_stem_engraver`. Care must be taken to ensure that automatic beams do not beam the notes on one staff when it’s not required on the other.

```
\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{
  \new PianoStaff <<
    \new Staff {
      <b d'>4 r d'16\> e'8. g8 r\!
      e'8 f' g'4 e'2
    }
    \new Staff {
      \clef bass
```

```
      \voiceOne
      \autoBeamOff
      \crossStaff { <e g>4 e, g16 a8. c8} d
      \autoBeamOn
      g8 f g4 c2
    }
  >>
}
```



For the time being, this engraver can not be specified by its name in double quotes, but rather prefixing its name with a hash symbol `#`, due to the way it is implemented.

Frammenti di codice selezionati

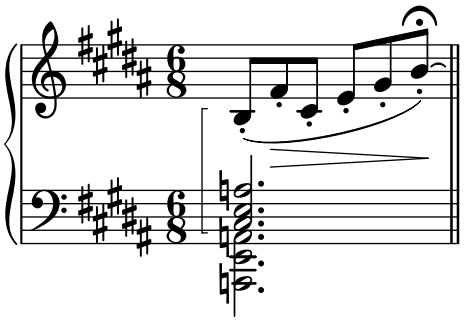
Indicating cross-staff chords with arpeggio bracket

An arpeggio bracket can indicate that notes on two different staves are to be played with the same hand. In order to do this, the `PianoStaff` must be set to accept cross-staff arpeggios and the arpeggios must be set to the bracket shape in the `PianoStaff` context.

(Debussy, Les collines d'Anacapri, m. 65)

```
\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \override PianoStaff.Arpeggio.stencil = #ly:arpeggio::brew-chord-bracket
  \new Staff {
    \relative c' {
      \key b \major
      \time 6/8
      b8-.(\arpeggio fis'-.\> cis-. e-. gis-. b-.)\!\fermata^\laissezVibrer
      \bar "||"
    }
  }
  \new Staff {
    \relative c' {
      \clef bass
      \key b \major
      <<
        {
          <a e cis>2.\arpeggio
        }
        \\
        {
          <a, e a,>2.
        }
      >>
    }
```

```
  }
>>
```



Vedi anche

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “Stem” in *Guida al Funzionamento Interno*.
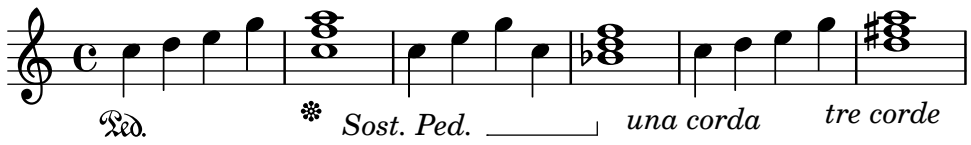
2.2.2 Piano

This section discusses notation issues that relate most directly to the piano.

Piano pedals

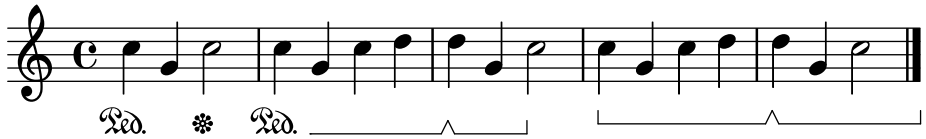
Pianos generally have three pedals that alter the way sound is produced: *sustain*, *sostenuto* (*sos.*), and *una corda* (*U.C.*). Sustain pedals are also found on vibraphones and celestas.

```
c4\sustainOn d e g
<c, f a>1\sustainOff
c4\sostenutoOn e g c,
<bes d f>1\sostenutoOff
c4\unaCorda d e g
<d fis a>1\treCorde
```



There are three styles of pedal indications: text, bracket, and mixed. The sustain pedal and the una corda pedal use the text style by default while the sostenuto pedal uses mixed by default.

```
c4\sustainOn g c2\sustainOff
\set Staff.pedalSustainStyle = #'mixed
c4\sustainOn g c d
d\sustainOff\sustainOn g, c2\sustainOff
\set Staff.pedalSustainStyle = #'bracket
c4\sustainOn g c d
d\sustainOff\sustainOn g, c2
\bar "|."
```



The placement of the pedal commands matches the physical movement of the sustain pedal during piano performance. Pedalling to the final bar line is indicated by omitting the final pedal off command.

Pedal indications may be placed in a `Dynamics` context, which aligns them on a horizontal line.

Vedi anche

Notation Reference: ⟨undefined⟩ [Ties], pagina ⟨undefined⟩.

Snippets: Sezione "Keyboards" in *Frammenti di codice*.

Internals Reference: Sezione "SustainPedal" in *Guida al Funzionamento Interno*, Sezione "SustainPedalLineSpanner" in *Guida al Funzionamento Interno*, Sezione "SustainEvent" in *Guida al Funzionamento Interno*, Sezione "SostenutoPedal" in *Guida al Funzionamento Interno*, Sezione "SostenutoPedalLineSpanner" in *Guida al Funzionamento Interno*, Sezione "SostenutoEvent" in *Guida al Funzionamento Interno*, Sezione "UnaCordaPedal" in *Guida al Funzionamento Interno*, Sezione "UnaCordaPedalLineSpanner" in *Guida al Funzionamento Interno*, Sezione "UnaCordaEvent" in *Guida al Funzionamento Interno*, Sezione "PianoPedalBracket" in *Guida al Funzionamento Interno*, Sezione "Piano_pedal_engraver" in *Guida al Funzionamento Interno*.

2.2.3 Accordion

This section discusses notation that is unique to the accordion.

Discant symbols

Accordions are often built with more than one set of reeds that may be in unison with, an octave above, or an octave below the written pitch. Each accordion maker has different names for the *shifts* that select the various reed combinations, such as *oboe*, *musette*, or *bandonium*, so a system of symbols has come into use to simplify the performance instructions.

Frammenti di codice selezionati

Accordion register symbols
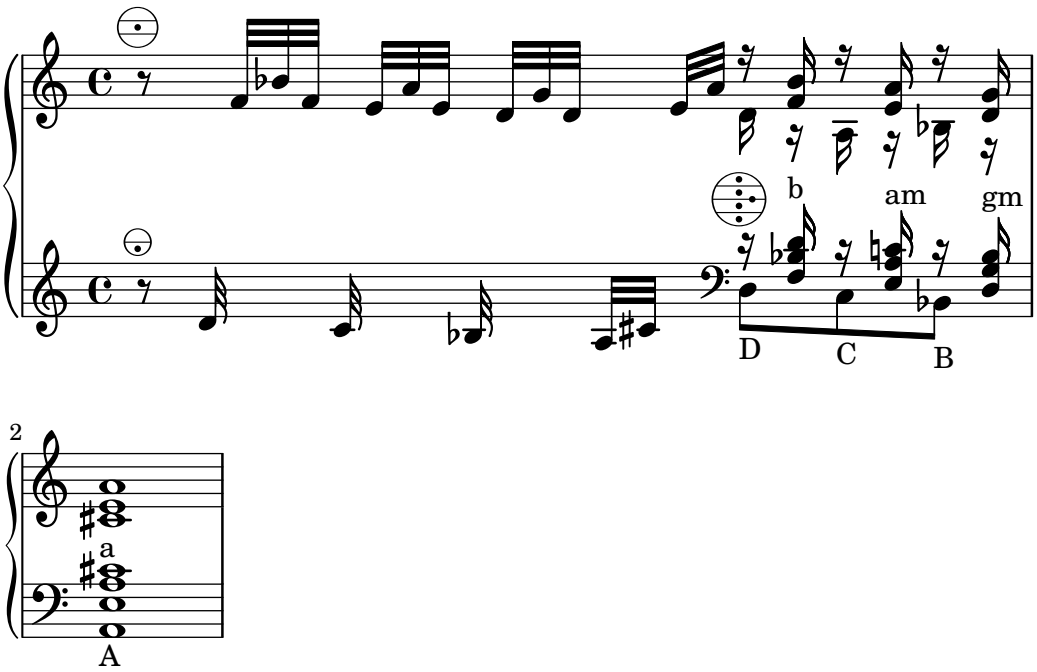
Accordion register symbols are available as `\markup` as well as as standalone music events (as register changes tend to occur between actual music events. Bass registers are not overly standardized. The available commands can be found in Sezione "Accordion Registers" in *Guida alla Notazione*.

```
\layout { ragged-right = ##t }

#(use-modules (scm accreg))

\new PianoStaff
<<
  \new Staff \relative
  { \clef treble \discant "10" r8 s32 f'[ bes f] s e[ a e] s d[ g d] s16 e32[ a]
    << { r16 <f bes> r <e a> r <d g> } \\ { d r a r bes r } >> | <cis e a>1 }
  \new Staff \relative
  { \clef treble \freeBass "1" r8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
    \clef bass \stdBass "Master"
    << { r16 <f, bes d>^"b" r <e a c>^"am" r <d g bes>^"gm" |
         <e a cis>1^"a" } \\
       { d8_"D" c_"C" bes_"B" | a1_"A" }
     >>
  }
```

>>



Vedi anche

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

2.2.4 Harp

This section discusses notation issues that are unique to the harp.

References for harps

Some common characteristics of harp music are covered elsewhere:

- The glissando is the most characteristic harp technique, [Glissando], pagina 133.
- A *bisbigliando* is written as a tremelo ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩.
- Natural harmonics are covered under [Harmonics], pagina 322.
- For directional arpeggios and non-arpeggios, see [Arpeggio], pagina 137.

Vedi anche
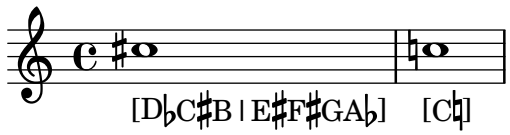
Notation Reference: ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩, [Glissando], pagina 133, [Arpeggio], pagina 137, [Harmonics], pagina 322.

Harp pedals

Harps have seven strings per octave that may be sounded at the natural, flattened, or sharpened pitch. In lever harps, each string is adjusted individually, but in pedal harps every string with the same pitch name is controlled by a single pedal. From the player’s left to right, the pedals are D, C, and B on the left and E, F, G, and A on the right. The position of the pedals may be indicated with text marks:

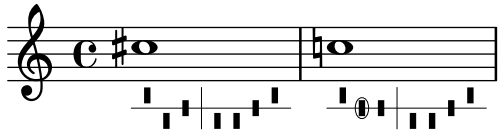
```
\textLengthOn
cis1_\markup \concat \vcenter {
  [D \flat C \sharp B|E \sharp F \sharp G A \flat] }
c!1_\markup \concat \vcenter {
```

```
[ C \natural ] }
```



or pedal diagrams:

```
\textLengthOn
cis1_\markup { \harp-pedal #"^v-|vv-^" }
c!1_\markup { \harp-pedal #"^o--|vv-^" }
```

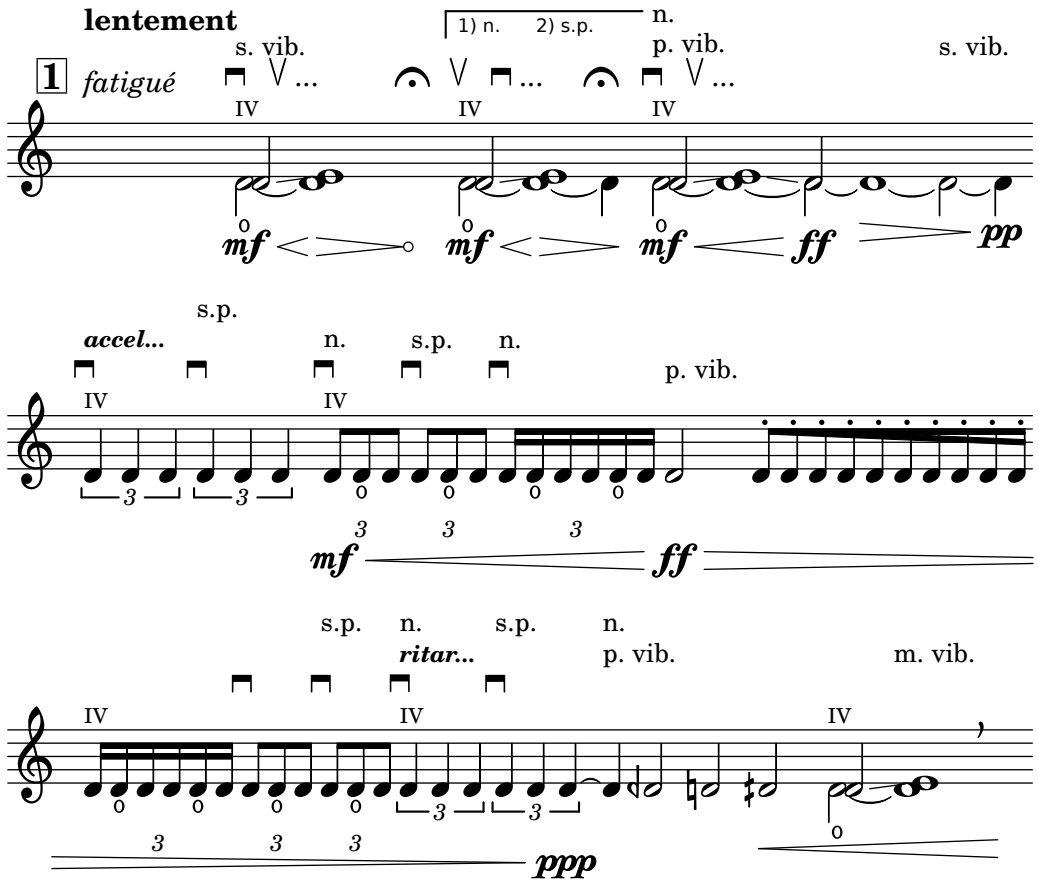


The `\harp-pedal` command accepts a string of characters, where `^` is the highest pedal position (flattened pitch), `-` is the middle pedal position (natural pitch), `v` is the lowest pedal position (sharpened pitch), and `|` is the divider. A prefixed `o` will circle the following pedal symbol.

Vedi anche

Notation Reference: ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩, Sezione A.10.5 [Instrument Specific Markup], pagina 689.

2.3 Unfretted string instruments



This section provides information and references which are helpful when writing for unfretted string instruments, principally orchestral strings.

2.3.1 Common notation for unfretted strings

There is little specialist notation for unfretted string instruments. The music is notated on a single staff, and usually only a single voice is required. Two voices might be required for some double-stopped or divisi passages.

References for unfretted strings

Most of the notation which is useful for orchestral strings and other bowed instruments is covered elsewhere:

- Textual indications such as "pizz." and "arco" are added as simple text – see ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩.
- Fingerings, including the thumb indication, are described in ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩.
- Double stopping is normally indicated by writing a chord, see ⟨undefined⟩ [Chorded notes], pagina ⟨undefined⟩. Directives for playing chords may be added, see [Arpeggio], pagina 137.
- Templates for string quartets can be found in Sezione "String quartet templates" in *Manuale di Apprendimento*. Others are shown in the snippets.

Vedi anche

Learning Manual: Sezione "String quartet templates" in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩, ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩, ⟨undefined⟩ [Chorded notes], pagina ⟨undefined⟩, [Arpeggio], pagina 137.
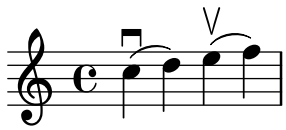
Snippets: Sezione "Unfretted strings" in *Frammenti di codice*.

Bowing indications

Bowing indications are created as articulations, which are described in ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.
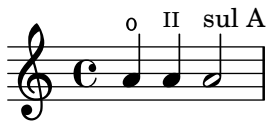
The bowing commands, `\upbow` and `\downbow`, are used with slurs as follows:

```
c4(\downbow d) e(\upbow f)
```



and the following example shows three ways in which an open A string on a violin might be indicated:

```
a4 \open
a^\markup { \teeny "II" }
a2^\markup { \small "sul A" }
```



Comandi predefiniti

`\downbow`, `\upbow`, `\open`.

Vedi anche
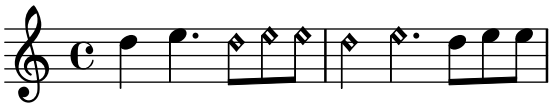
Notation Reference: ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩, ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩.

Harmonics

Natural harmonics

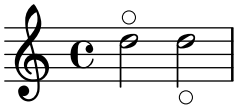
Natural harmonics can be notated in several ways. A diamond-shaped note head generally means to touch the string where you would stop the note if it were not a diamond.

```
d4 e4.
\harmonicsOn
d8 e e
d4 e4.
\harmonicsOff
d8 e e
```



Alternatively a normal note head is shown at the pitch to be sounded together with a small circle to indicate it should be played as a harmonic:

```
d2^\flageolet d_\flageolet
```
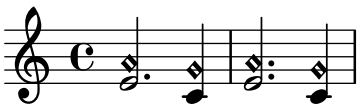


A smaller circle may be created, see the snippet list in [References for unfretted strings], pagina 321.

Artificial harmonics

Artificial harmonics are notated with two notes, one with a normal note head indicating the stopped position and one with an open diamond note head to indicate the harmonic position.

Artificial harmonics indicated with `\harmonic` do not show the dots. The context property `harmonicDots` should be set if dots are required.

```
<e a\harmonic>2.  <c g'\harmonic>4
\set harmonicDots = ##t
<e a\harmonic>2.  <c g'\harmonic>4
```



Nota: `\harmonic` **must** be placed inside a chord construct even if there is only a single note. Normally `\harmonicsOn` would be used in this situation.

Vedi anche

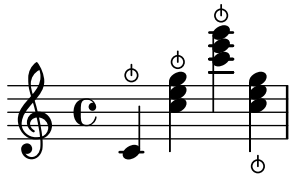
Music Glossary: Sezione “harmonics” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Special note heads], pagina ⟨undefined⟩, [References for unfretted strings], pagina 321.
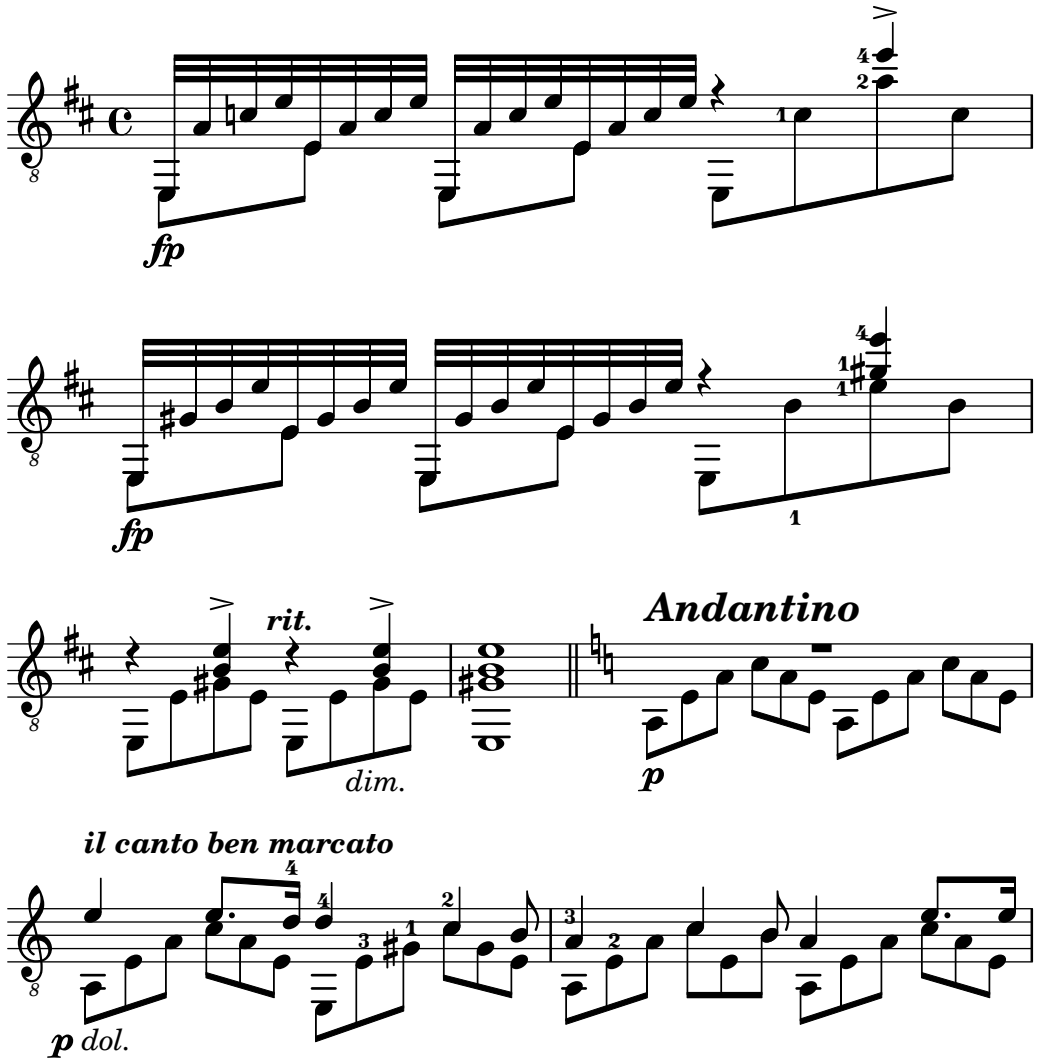
Snap (Bartók) pizzicato
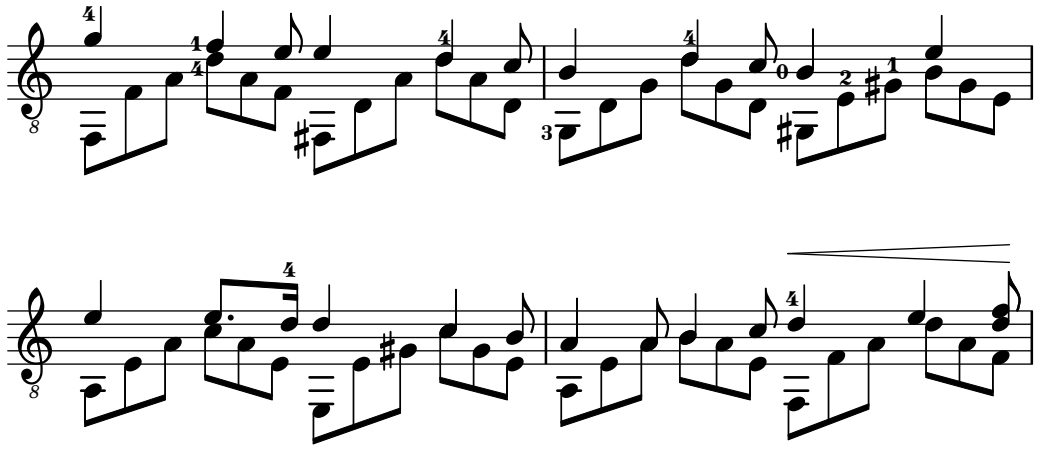
A *snap pizzicato* (also known as “Bartok pizz”) is a type of pizzicato where the string is deliberately plucked upwards (rather than sideways) such that it hits the fingerboard.

```
c4\snappizzicato
<c' e g>4\snappizzicato
<c' e g>4^\snappizzicato
<c, e g>4_\snappizzicato
```



2.4 Fretted string instruments





This section discusses several aspects of music notation that are unique to fretted string instruments.

2.4.1 Common notation for fretted strings

This section discusses common notation that is unique to fretted string instruments.

References for fretted strings

Music for fretted string instruments is normally notated on a single staff, either in traditional music notation or in tablature. Sometimes the two types are combined, and it is especially common in popular music to use chord diagrams above a staff of traditional notation. The guitar and the banjo are transposing instruments, sounding an octave lower than written. Scores for these instruments should use the `"treble_8"` clef (or `\transposition c` to get correct MIDI output). Some other elements pertinent to fretted string instruments are covered elsewhere:

- Fingerings are indicated as shown in ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩.
- Instructions for *Laissez vibrer* ties as well as ties on arpeggios and tremolos can be found in ⟨undefined⟩ [Ties], pagina ⟨undefined⟩.
- Instructions for handling multiple voices can be found in ⟨undefined⟩ [Collision resolution], pagina ⟨undefined⟩.
- Instructions for indicating harmonics can be found in [Harmonics], pagina 322.

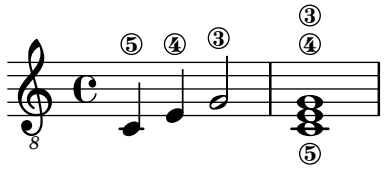
Vedi anche

Notation Reference: ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩, ⟨undefined⟩ [Ties], pagina ⟨undefined⟩, ⟨undefined⟩ [Collision resolution], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument names], pagina ⟨undefined⟩, ⟨undefined⟩ [Writing music in parallel], pagina ⟨undefined⟩, [Arpeggio], pagina 137, Sezione A.13 [List of articulations], pagina 706, ⟨undefined⟩ [Clef], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩.

String number indications

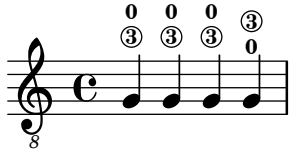
The string on which a note should be played may be indicated by appending `\number` to a note.

```
\clef "treble_8"
c4\5 e\4 g2\3
<c,\5 e\4 g\3>1
```



When fingerings and string indications are used together, their placement can be controlled by the order in which the two items appear in the code *only* if they appear inside of an explicit chord: applied to whole chords or single notes *outside* of chords, fingerings are placed using a different mechanism.

```
\clef "treble_8"
g4\3-0
g-0\3
<g\3-0>
<g-0\3>
```

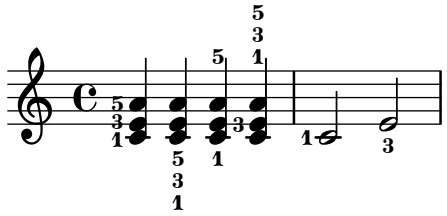


Frammenti di codice selezionati

Controllare il posizionamento delle diteggiature di un accordo

Il posizionamento dei numeri della diteggiatura può essere regolato in modo preciso. Perché l'orientamento funzioni, occorre usare il costrutto per gli accordi `<>` anche per una nota singola.

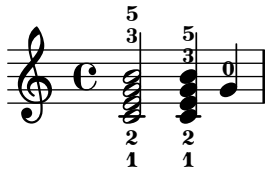
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Far sì che la diteggiatura appaia dentro il rigo

Per impostazione predefinita, le diteggiature orientate verticalmente sono poste fuori dal rigo. Tuttavia, questo comportamento può essere annullato. Attenzione: bisogna usare il costrutto per gli accordi `<>`, anche se si riferisce a una singola nota.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 <g'-0>
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

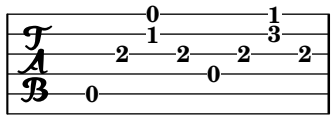
Internals Reference: Sezione “StringNumber” in *Guida al Funzionamento Interno*, Sezione “Fingering” in *Guida al Funzionamento Interno*.

Default tablatures

Music for plucked string instruments is frequently notated using a finger/touch notation or tablature. In contrast to traditional notation pitches are not denoted with note heads, but by numbers (or letter-like symbols in historical intavolatura). The staff lines in tablature indicate the string on which the note is to be played, and a number placed on a staff line indicated the fret at which the corresponding string is to be pressed. Notes that are to be played simultaneously are vertically aligned.

By default, string 1 is the highest string, and corresponds to the top line on the `TabStaff`. The tuning of the `TabStaff` strings defaults to the standard guitar tuning (with 6 strings). The notes are printed as tablature, by using `TabStaff` and `TabVoice` contexts. A calligraphic tablature clef is added automatically.

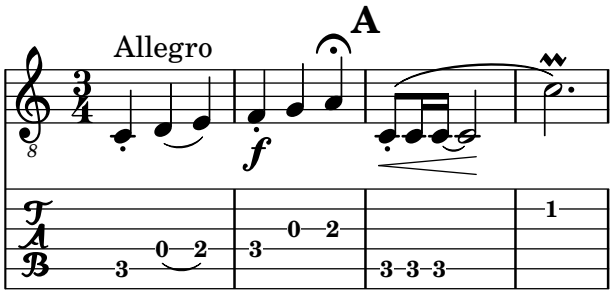
```
\new TabStaff \relative c' {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```



Default tablatures do not contain any symbols for tone duration nor any other musical symbols such as e.g. expressive marks.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ c2\!
  c'2.\prall\)
}

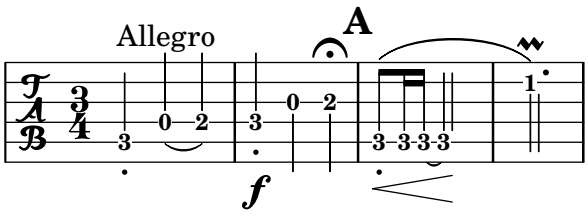
\score {
  <<
    \new Staff { \clef "G_8" \symbols }
    \new TabStaff { \symbols }
  >>
}
```



If all musical symbols used in traditional notation should also show up in tablature one has to apply the command `\tabFullNotation` in a `TabStaff`-context. Please bear in mind that half notes are double-stemmed in tablature in order to distinguish them from quarter notes.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ c2\!
  c'2.\prall\)
}

\score {
  \new TabStaff {
    \tabFullNotation
    \symbols
  }
}
```

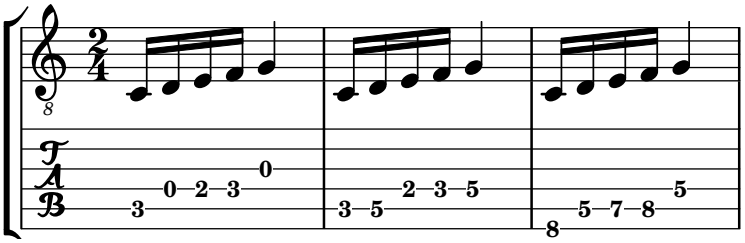


By default pitches are assigned to the lowest playing position on the fret-board (first position). Open strings are automatically preferred. If you would like a certain pitch to be played on a specific string you can add a string number indication to the pitch name. If you don't want to have string number indications appear in traditional notation, you can override the respective stencil. Usually it will be more comfortable to define the playing position by using the value of `minimumFret`. The default value for minimumFret is 0.

Even when `minimumFret` is set, open strings are used whenever possible. This behaviour can be changed by setting `restrainOpenStrings` to `#t`.

```
\layout { \omit Voice.StringNumber }
\new StaffGroup <<
   \new Staff \relative c {
     \clef "treble_8"
     \time 2/4
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     c,16 d e f g4
   }
   \new TabStaff \relative c {
     c16 d e f g4
```

```
    c,16\5 d\5 e\4 f\4 g4\4
    \set TabStaff.minimumFret = #5
    \set TabStaff.restrainOpenStrings = ##t
    c,16 d e f g4
  }
>>
```



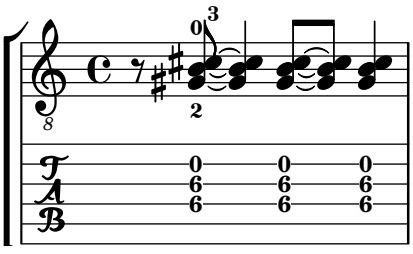
Chord constructs can be repeated by the chord repetition symbol `q`. In combination with tabulatures, its behavior of removing string and finger numbers alongside with other events is cumbersome, so you'll want to run

```
\chordRepeats #'(string-number-event fingering-event)
```

explicitly on music expressions in tabulature using ⟨undefined⟩ [Chord repetition], pagina ⟨undefined⟩. This particular command is so common that it is available as `\tabChordRepeats`.

```
guitar = \relative c' {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ q q4
}

\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \guitar
  }
  \new TabStaff {
    \tabChordRepeats \guitar
  }
>>
```

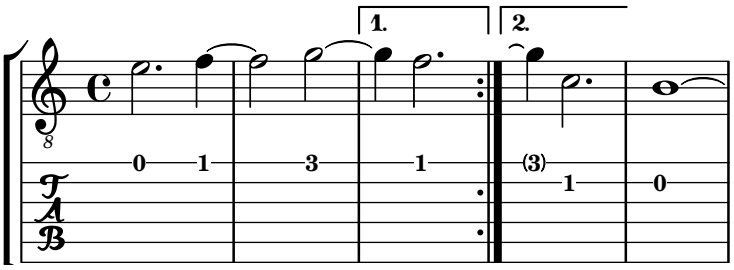


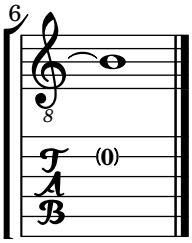
Ties over a line break are parenthesized by default. The same holds for the second alternative of a repeat.

```
ties = \relative c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~
  }
  \alternative {
     { g4 f2. }
```

```
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```



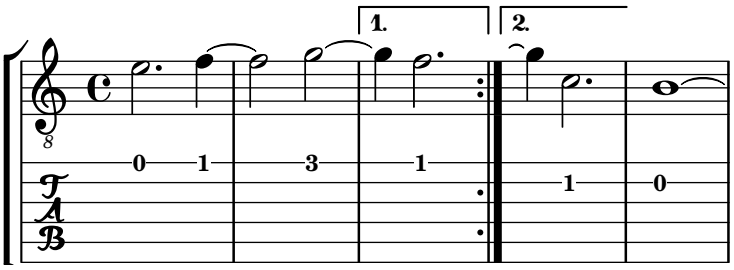


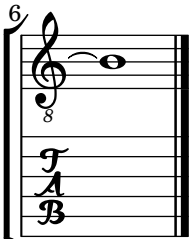
The command `\hideSplitTiedTabNotes` cancels the behavior of engraving fret numbers in parentheses:

```
ties = \relative c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~ }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
```

```
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
      \hideSplitTiedTabNotes
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```

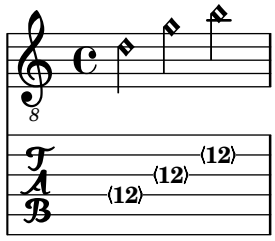




Harmonic indications can be added to tablature notation as sounding pitches:

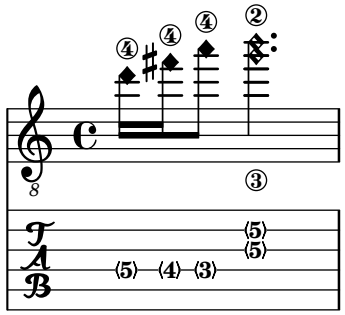
```
\layout { \omit Voice.StringNumber }
firstHarmonic = {
  d'4\4\harmonic
  g'4\3\harmonic
  b'2\2\harmonic
}
\score {
  <<
```

```
    \new Staff {
      \clef "treble_8"
      \firstHarmonic
    }
    \new TabStaff { \firstHarmonic }
  >>
}
```



Note that the command `\harmonic` must always be attached to single notes (possibly inside of a chord) instead of whole chords. It only makes sense for open-string harmonics in the 12th fret. All other harmonics should be calculated by LilyPond. This can be achieved by indicating the fret where a finger of the fretting hand should touch a string.

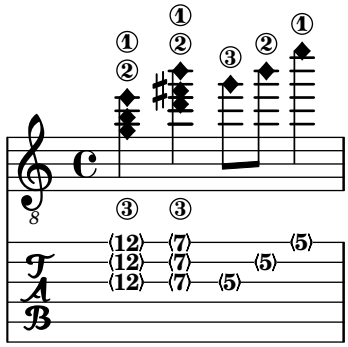
```
fretHarmonics = {
  \harmonicByFret #5 d16\4
  \harmonicByFret #4 d16\4
  \harmonicByFret #3 d8\4
  \harmonicByFret #5 <g\3 b\2>2.
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \fretHarmonics
    }
    \new TabStaff { \fretHarmonics }
  >>
}
```



Alternatively, harmonics can be computed by defining the ratio of string lengths above and below the harmonic fingering.

```
ratioHarmonics = {
  \harmonicByRatio #1/2 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/3 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }
```

```
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \ratioHarmonics
    }
    \new TabStaff { \ratioHarmonics }
  >>
}
```

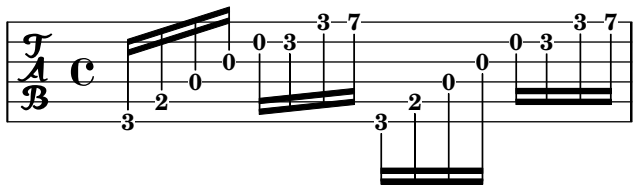


Frammenti di codice selezionati

Stem and beam behavior in tablature

The direction of stems is controlled the same way in tablature as in traditional notation. Beams can be made horizontal, as shown in this example.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = #10000
    g,,16 b d g b d g b
  }
}
```



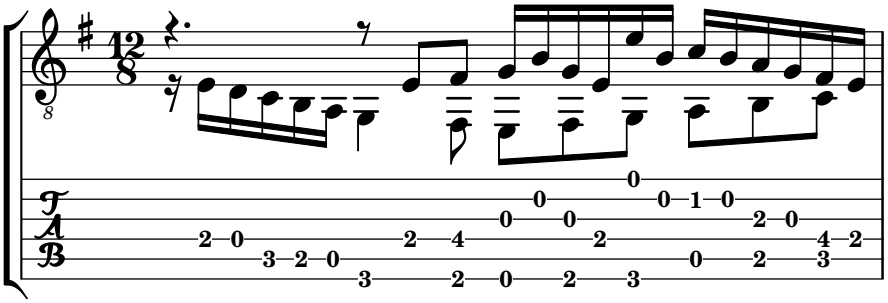
Polyphony in tablature

Polyphony is created the same way in a `TabStaff` as in a regular staff.

```
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}
```

```
lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  <<
    \new StaffGroup = "tab with traditional" <<
      \new Staff = "guitar traditional" <<
        \clef "treble_8"
        \context Voice = "upper" \upper
        \context Voice = "lower" \lower
      >>
      \new TabStaff = "guitar tab" <<
        \context TabVoice = "upper" \upper
        \context TabVoice = "lower" \lower
      >>
    >>
  >>
}
```



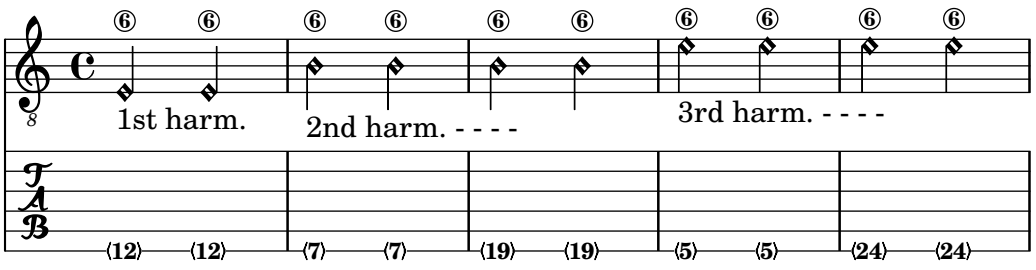
Open string harmonics in tablature

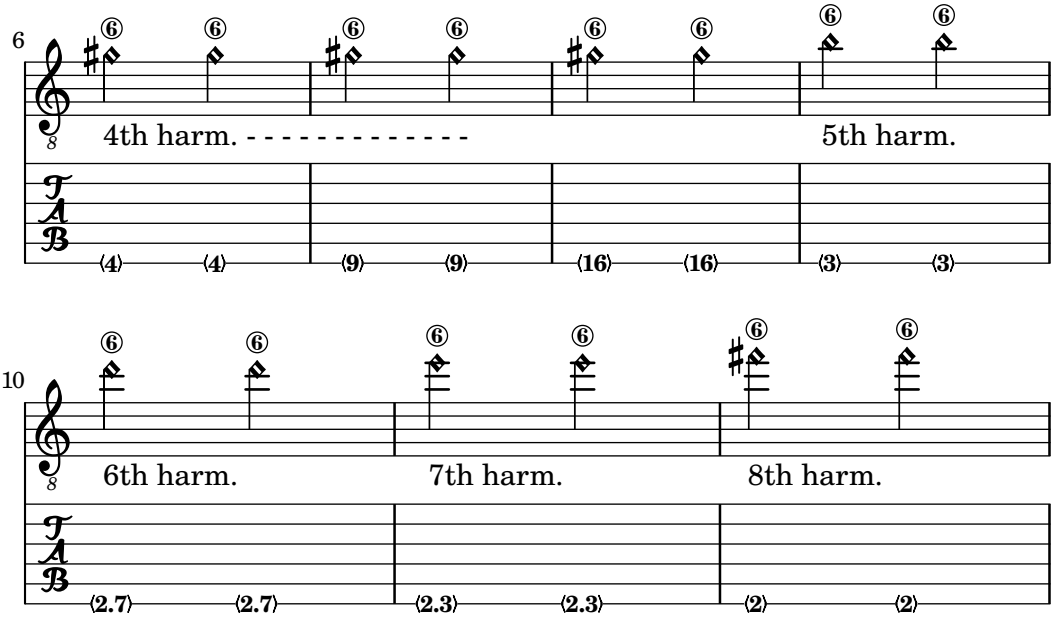
This snippet demonstrates open-string harmonics

```
openStringHarmonics = {
  %first harmonic
  \harmonicByFret #12 e,2\6_\markup{"1st harm."}
  \harmonicByRatio #1/2 e,\6
  %second harmonic
  \harmonicByFret #7 e,\6_\markup{"2nd harm. - - - -"}
  \harmonicByRatio #1/3 e,\6
  \harmonicByFret #19 e,\6
  \harmonicByRatio #2/3 e,\6
  %\harmonicByFret #19 < e,\6 a,\5 d\4 >
  %\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >
  %third harmonic
  \harmonicByFret #5 e,\6_\markup{"3rd harm. - - - -"}
  \harmonicByRatio #1/4 e,\6
  \harmonicByFret #24 e,\6
  \harmonicByRatio #3/4 e,\6
  \break
```

```
  %fourth harmonic
  \harmonicByFret #4 e,\6_\markup{"4th harm. - - - - - - - - - - - - -"}
  \harmonicByRatio #1/5 e,\6
  \harmonicByFret #9 e,\6
  \harmonicByRatio #2/5 e,\6
  \harmonicByFret #16 e,\6
  \harmonicByRatio #3/5 e,\6
  %fifth harmonic
  \harmonicByFret #3 e,\6_\markup{"5th harm."}
  \harmonicByRatio #1/6 e,\6
  \break
  %sixth harmonic
  \harmonicByFret #2.7 e,\6_\markup{"6th harm."}
  \harmonicByRatio #1/7 e,\6
  %seventh harmonic
  \harmonicByFret #2.3 e,\6_\markup{"7th harm."}
  \harmonicByRatio #1/8 e,\6
  %eighth harmonic
  \harmonicByFret #2 e,\6_\markup{"8th harm."}
  \harmonicByRatio #1/9 e,\6
}

\score {
  <<
    \new Staff {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}
```





Fretted-string harmonics in tablature

Demonstrates fretted-string harmonics in tablature

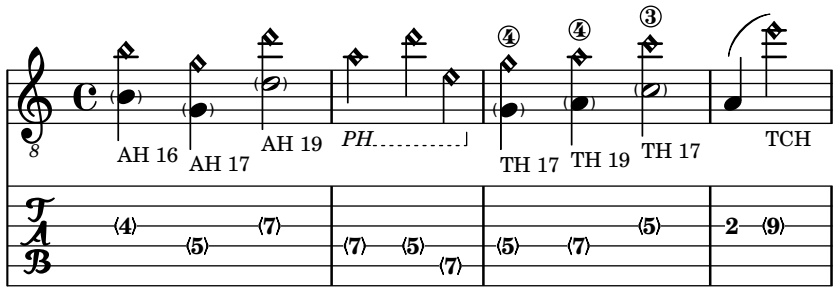
```
pinchedHarmonics = {
   \textSpannerDown
   \override TextSpanner.bound-details.left.text =
      \markup {\halign #-0.5 \teeny "PH" }
      \override TextSpanner.style =
         #'dashed-line
   \override TextSpanner.dash-period = #0.6
   \override TextSpanner.bound-details.right.attach-dir = #1
   \override TextSpanner.bound-details.right.text =
      \markup { \draw-line #'(0 . 1) }
   \override TextSpanner.bound-details.right.padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4_\markup{ \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4_\markup{ \teeny "AH 17" }
  <\parenthesize d' d''\harmonic>2_\markup{ \teeny "AH 19" }
  %pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <d''\harmonic>4
  <e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4_\markup{ \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4_\markup{ \teeny "TH 19" }
  <\parenthesize c'\3 c''\harmonic>2_\markup{ \teeny "TH 17" }
  %touch harmonics (TCH)
  a4( <e''\harmonic>2. )_\markup{ \teeny "TCH" }
}

frettedStrings = {
  %artificial harmonics (AH)
```

```
  \harmonicByFret #4 g4\3
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 g2\3
  %pinched harmonics (PH)
  \harmonicByFret #7 d2\4
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 a4\5
  %tapped harmonics (TH)
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 d4\4
  \harmonicByFret #5 g2\3
  %touch harmonics (TCH)
  a4 \harmonicByFret #9 g2.\3
}

\score {
  <<
    \new Staff {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}
```



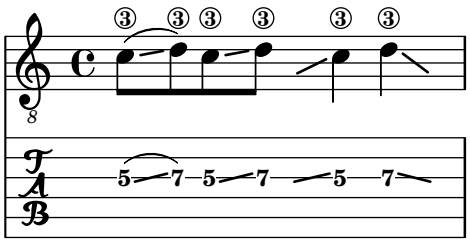
Slides in tablature

Slides can be typeset in both `Staff` and `TabStaff` contexts:

```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
  \stemDown \hideNotes
```

```
  g16 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>
  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = #4
      \override Glissando.springs-and-rods =
                          #ly:spanner::set-spacing-rods
      \override Glissando.thickness = #2
    }
  }
}
```

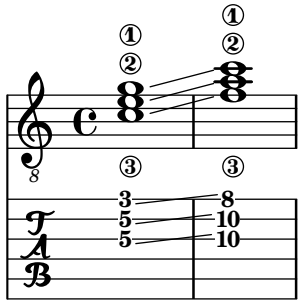


Chord glissando in tablature

Slides for chords can be indicated in both `Staff` and `TabStaff`. String numbers are necessary for `TabStaff` because automatic string calculations are different for chords and for single notes.

```
myMusic = \relative c' {
  <c\3 e\2 g\1>1 \glissando <f\3 a\2 c\1>
}

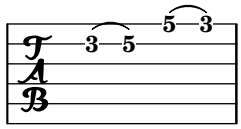
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff {
      \myMusic
    }
  >>
}
```



Hammer on and pull off

Hammer-on and pull-off can be obtained using slurs.

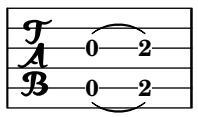
```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



Hammer on and pull off using voices

The arc of hammer-on and pull-off is upwards in voices one and three and downwards in voices two and four:

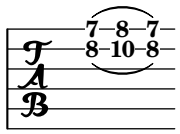
```
\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}
```



Hammer on and pull off using chords

When using hammer-on or pull-off with chorded notes, only a single arc is drawn. However 'double arcs' are possible by setting the `doubleSlurs` property to `#t`.

```
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    <g' b>8( <a c> <g b>)
  }
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Chord repetition], pagina ⟨undefined⟩, [Glissando], pagina 133, [Harmonics], pagina 322, ⟨undefined⟩ [Stems], pagina ⟨undefined⟩, ⟨undefined⟩ [Written-out repeats], pagina ⟨undefined⟩.

Snippets: Sezione "Fretted strings" in *Frammenti di codice*.

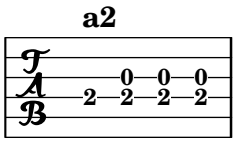
Internals Reference: Sezione "TabNoteHead" in *Guida al Funzionamento Interno*, Sezione "TabStaff" in *Guida al Funzionamento Interno*, Sezione "TabVoice" in *Guida al Funzionamento Interno*, Sezione "Beam" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Chords are not handled in a special way, and hence the automatic string selector may easily select the same string for two notes in a chord.

In order to handle `\partcombine`, a `TabStaff` must use specially-created voices:

```
melodia = \partcombine { e4 g g g } { e4 e e e }
<<
  \new TabStaff <<
    \new TabVoice = "one" s1
    \new TabVoice = "two" s1
    \new TabVoice = "shared" s1
    \new TabVoice = "solo" s1
    { \melodia }
  >>
>>
```



Guitar special effects are limited to harmonics and slides.

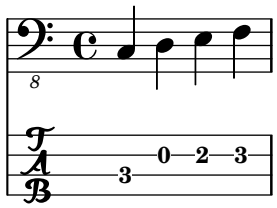
Custom tablatures

LilyPond tablature automatically calculates the fret for a note based on the string to which the note is assigned. In order to do this, the tuning of the strings must be specified. The tuning of the strings is given in the `stringTunings` property.

LilyPond comes with predefined string tunings for banjo, mandolin, guitar, bass guitar, ukulele, violin, viola, cello, and double bass. LilyPond automatically sets the correct transposition for predefined tunings. The following example is for bass guitar, which sounds an octave lower than written.

```
<<
  \new Voice \with {
    \omit StringNumber
  } {
    \clef "bass_8"
    \relative c, {
      c4 d e f
    }
  }
  \new TabStaff \with {
    stringTunings = #bass-tuning
```

```
  } {
    \relative c, {
      c4 d e f
    }
  }
>>
```



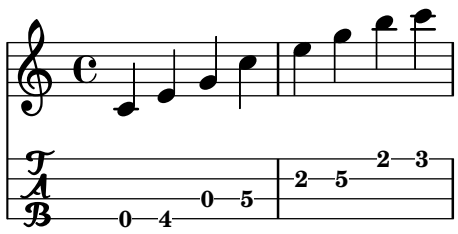
The default string tuning is `guitar-tuning`, which is the standard EADGBE tuning. Some other predefined tunings are `guitar-open-g-tuning`, `mandolin-tuning` and `banjo-open-g-tuning`. The predefined string tunings are found in '`ly/string-tunings-init.ly`'.

Any desired string tuning can be created. The `\stringTuning` function can be used to define a string tuning which can be used to set `stringTunings` for the current context.

Its argument is a chord construct defining the pitches of each string in the tuning. The chord construct must be in absolute octave mode, see ⟨undefined⟩ [Absolute octave entry], pagina ⟨undefined⟩. The string with the highest number (generally the lowest string) must come first in the chord. For example, we can define a string tuning for a four-string instrument with pitches of `a''`, `d''`, `g'`, and `c'`:

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set Staff.stringTunings = \stringTuning <c' g' d'' a''>
    \mynotes
  }
>>
```



The `stringTunings` property is also used by `FretBoards` to calculate automatic fret diagrams.

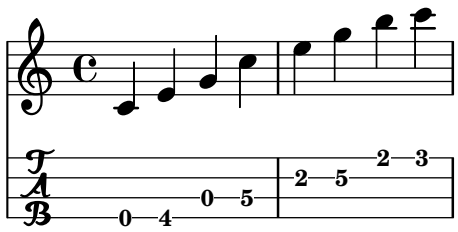
String tunings are used as part of the hash key for predefined fret diagrams (see [Predefined fret diagrams], pagina 351).

The previous example could also be written as follows:

```
custom-tuning = \stringTuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #custom-tuning
    \mynotes
  }
>>
```



Internally, a string tuning is a Scheme list of string pitches, one for each string, ordered by string number from 1 to N, where string 1 is at the top of the tablature staff and string N is at the bottom. This ordinarily results in ordering from highest pitch to lowest pitch, but some instruments (e.g. ukulele) do not have strings ordered by pitch.

A string pitch in a string tuning list is a LilyPond pitch object. Pitch objects are created with the Scheme function `ly:make-pitch` (see Sezione A.21 [Scheme functions], pagina 754).

`\stringTuning` creates such an object from chord input.

LilyPond automatically calculates the number of lines in the `TabStaff` and the number of strings in an automatically calculated `FretBoard` as the number of elements in `stringTunings`.

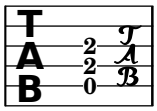
To let all TabStaff contexts use the same custom tuning by default, you can use

```
\layout {
  \context {
    \TabStaff
    stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```

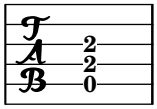
A modern tab clef can also be used.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
```

```
  <a, e a>1
}
```



2



The modern tab clef supports tablatures from 4 to 7 strings.

Vedi anche

Notation Reference: ⟨undefined⟩ [Absolute octave entry], pagina ⟨undefined⟩, [Predefined fret diagrams], pagina 351, Sezione A.21 [Scheme functions], pagina 754.

Installed Files: ‘`ly/string-tunings-init.ly`’, ‘`scm/tablature.scm`’.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “Tab_note_heads_engraver” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Automatic tablature calculations do not work properly in most cases for instruments where string pitches do not vary monotonically with string number, such as ukuleles.

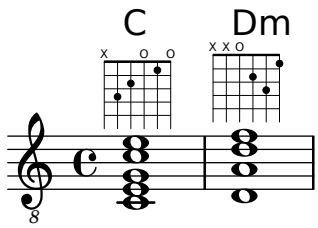
Fret diagram markups

Fret diagrams can be added to music as a markup to the desired note. The markup contains information about the desired fret diagram. There are three different fret-diagram markup interfaces: standard, terse, and verbose. The three interfaces produce equivalent markups, but have varying amounts of information in the markup string. Details about the syntax of the different markup strings used to define fret diagrams are found at Sezione A.10.5 [Instrument Specific Markup], pagina 689.

The standard fret diagram markup string indicates the string number and the fret number for each dot to be placed on the string. In addition, open and unplayed (muted) strings can be indicated.

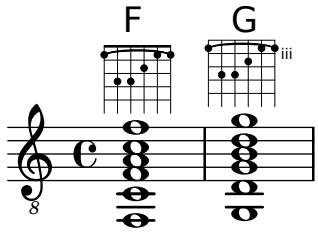
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram #"6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram #"6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
```

```
>>
```



Barre indications can be added to the diagram from the fret-diagram markup string.

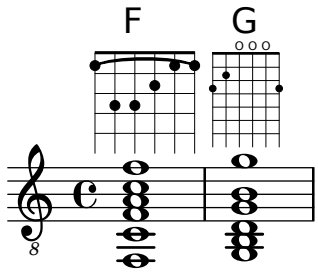
```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram #"c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram #"c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
    }
  }
>>
```



The size of the fret diagram, and the number of frets in the diagram can be changed in the fret-diagram markup string.

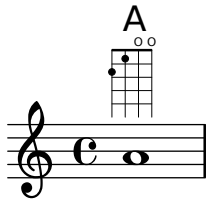
```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram #"s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram #"h:6;6-3;5-2;4-o;3-o;2-o;1-3;"
    }
  }
```

```
>>
```



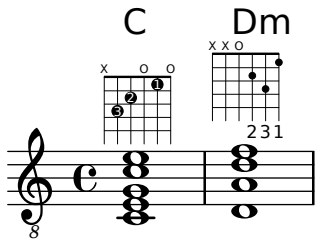
The number of strings in a fret diagram can be changed to accommodate different instruments such as banjos and ukuleles with the fret-diagram markup string.

```
<<
  \new ChordNames {
    \chordmode {
      a1
    }
  }
  \new Staff {
    % An 'A' chord for ukulele
    a'1^\markup {
      \fret-diagram #"w:4;4-2-2;3-1-1;2-o;1-o;"
    }
  }
>>
```



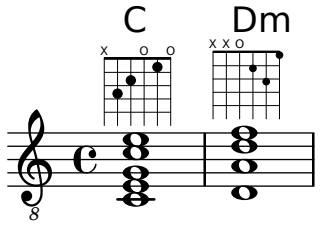
Fingering indications can be added, and the location of fingering labels can be controlled by the fret-diagram markup string.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram #"f:1;6-x;5-3-3;4-2-2;3-o;2-1-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram #"f:2;6-x;5-x;4-o;3-2-2;2-3-3;1-1-1;"
    }
  }
>>
```



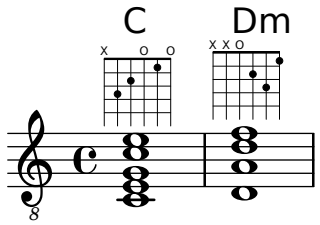
Dot radius and dot position can be controlled with the fret-diagram markup string.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram #"d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram #"p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



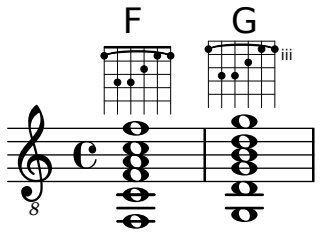
The fret-diagram-terse markup string omits string numbers; the string number is implied by the presence of semicolons. There is one semicolon for each string in the diagram. The first semicolon corresponds to the highest string number and the last semicolon corresponds to the first string. Mute strings, open strings, and fret numbers can be indicated.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse #"x;3;2;o;1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse #"x;x;o;2;3;1;"
    }
  }
>>
```



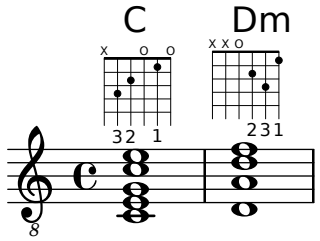
Barre indicators can be included in the fret-diagram-terse markup string.

```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram-terse #"1-(;3;3;2;1;1-);"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram-terse #"3-(;5;5;4;3;3-);"
    }
  }
>>
```



Fingering indications can be included in the fret-diagram-terse markup string.

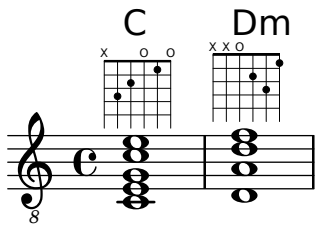
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse #"x;x;o;2-2;3-3;1-1;"
    }
  }
>>
```



Other fret diagram properties must be adjusted using `\override` when using the fret-diagram-terse markup.

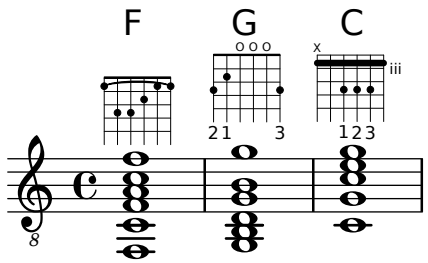
The fret-diagram-verbose markup string is in the format of a Scheme list. Each element of the list indicates an item to be placed on the fret diagram.

```
<<
    \new ChordNames {
      \chordmode {
        c1 d:m
      }
    }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (place-fret 5 3)
        (place-fret 4 2)
        (open 3)
        (place-fret 2 1)
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
>>
```



Fingering indications and barres can be included in a fret-diagram-verbose markup string. Unique to the fret-diagram-verbose interface is a capo indication that can be placed on the fret diagram. The capo indication is a thick bar that covers all strings. The fret with the capo will be the lowest fret in the fret diagram.

```
<<
    \new ChordNames {
      \chordmode {
        f1 g c
      }
    }
  \new Staff {
    \clef "treble_8"
    \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
    <f, c f a c' f'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 1)
        (place-fret 5 3)
        (place-fret 4 3)
        (place-fret 3 2)
        (place-fret 2 1)
        (place-fret 1 1)
        (barre 6 1 1)
      )
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 2)
        (place-fret 5 2 1)
        (open 4)
        (open 3)
        (open 2)
        (place-fret 1 3 3)
      )
    }
    <c g c' e' g'>1^\markup {
      \fret-diagram-verbose #'(
        (capo 3)
        (mute 6)
        (place-fret 4 5 1)
        (place-fret 3 5 2)
        (place-fret 2 5 3)
      )
    }
  }
>>
```



All other fret diagram properties must be adjusted using `\override` when using the fret-diagram-verbose markup.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*. For a fret diagram markup, the interface properties belong to `Voice.TextScript`.

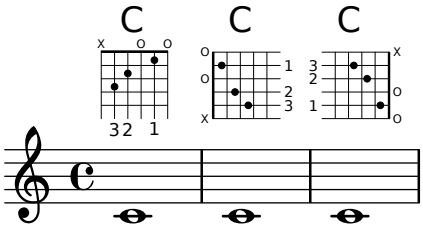
Frammenti di codice selezionati

Changing fret orientations

Fret diagrams can be oriented in three ways. By default the top string or fret in the different orientations will be aligned.

```
\include "predefined-guitar-fretboards.ly"

<<
  \chords {
    c1
    c1
    c1
  }
  \new FretBoards {
    \chordmode {
      c1
      \override FretBoard.fret-diagram-details.orientation =
        #'landscape
      c1
      \override FretBoard.fret-diagram-details.orientation =
        #'opposing-landscape
      c1
    }
  }
  \new Voice {
    c'1
    c'1
    c'
  }
>>
```



Customizing markup fret diagrams

Fret diagram properties can be set through `'fret-diagram-details`. For markup fret diagrams, overrides can be applied to the `Voice.TextScript` object or directly to the markup.

```
<<
  \chords { c1 | c | c | d }

  \new Voice = "mel" {
    \textLengthOn
    % Set global properties of fret diagram
```

```
\override TextScript.size = #'1.2
\override TextScript.fret-diagram-details.finger-code = #'in-dot
\override TextScript.fret-diagram-details.dot-color = #'white

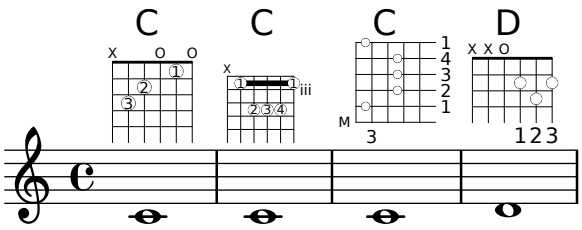
%% C major for guitar, no barre, using defaults
   % terse style
c'1^\markup { \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;" }

%% C major for guitar, barred on third fret
   % verbose style
   % size 1.0
   % roman fret label, finger labels below string, straight barre
c'1^\markup {
  % standard size
  \override #'(size . 1.0) {
    \override #'(fret-diagram-details . (
                 (number-type . roman-lower)
                 (finger-code . in-dot)
                 (barre-type . straight))) {
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1)
                               (place-fret 4 5 2)
                               (place-fret 3 5 3)
                               (place-fret 2 5 4)
                               (place-fret 1 3 1)
                               (barre 5 1 3))
    }
  }
}

%% C major for guitar, barred on third fret
   % verbose style
   % landscape orientation, arabic numbers, M for mute string
   % no barre, fret label down or left, small mute label font
c'1^\markup {
  \override #'(fret-diagram-details . (
               (finger-code . below-string)
               (number-type . arabic)
               (label-dir . -1)
               (mute-string . "M")
               (orientation . landscape)
               (barre-type . none)
               (xo-font-magnification . 0.4)
               (xo-padding . 0.3))) {
    \fret-diagram-verbose #'((mute 6)
                             (place-fret 5 3 1)
                             (place-fret 4 5 2)
                             (place-fret 3 5 3)
                             (place-fret 2 5 4)
                             (place-fret 1 3 1)
                             (barre 5 1 3))
  }
```

```
    }

    %% simple D chord
       % terse style
       % larger dots, centered dots, fewer frets
       % label below string
    d'1^\markup {
      \override #'(fret-diagram-details . (
                    (finger-code . below-string)
                    (dot-radius . 0.35)
                    (dot-position . 0.5)
                    (fret-count . 3))) {
        \fret-diagram-terse #"x;x;o;2-1;3-2;2-3;"
      }
    }
  }
>>
```



Vedi anche

Notation Reference: Sezione A.10.5 [Instrument Specific Markup], pagina 689.

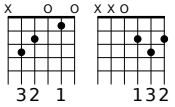
Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

Predefined fret diagrams

Fret diagrams can be displayed using the `FretBoards` context. By default, the `FretBoards` context will display fret diagrams that are stored in a lookup table:

```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {
    c1 d
  }
}
```



The default predefined fret diagrams are contained in the file ‘`predefined-guitar-fretboards.ly`’. Fret diagrams are stored based on the pitches of a chord and the value of `stringTunings` that is currently in use. ‘`predefined-guitar-fretboards.ly`’ contains predefined fret diagrams only for `guitar-tuning`. Predefined fret diagrams can be added for other instruments or other tunings by following the examples found in ‘`predefined-guitar-fretboards.ly`’.

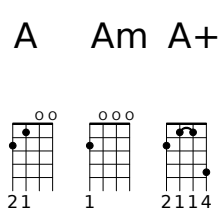
Fret diagrams for the ukulele are contained in the file ‘predefined-ukulele-fretboards.ly’.

```
\include "predefined-ukulele-fretboards.ly"

myChords = \chordmode { a1 a:m a:aug }

\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #ukulele-tuning
  \myChords
}
```



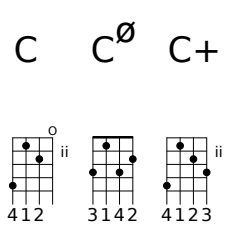
Fret diagrams for the mandolin are contained in the file ‘predefined-mandolin-fretboards.ly’.

```
\include "predefined-mandolin-fretboards.ly"

myChords = \chordmode { c1 c:m7.5- c:aug }

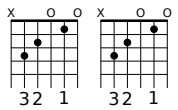
\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}
```



Chord pitches can be entered either as simultaneous music or using chord mode (see [Chord mode overview], pagina 392).

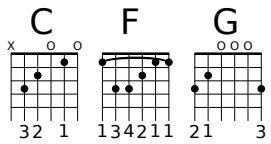
```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode { c1 }
  <c' e' g'>1
}
```



It is common that both chord names and fret diagrams are displayed together. This is achieved by putting a `ChordNames` context in parallel with a `FretBoards` context and giving both contexts the same music.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}

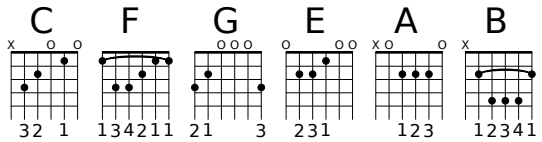
<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



Predefined fret diagrams are transposable, as long as a diagram for the transposed chord is stored in the fret diagram table.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}

mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \new ChordNames {
    \mychordlist
  }
  \new FretBoards {
    \mychordlist
  }
>>
```

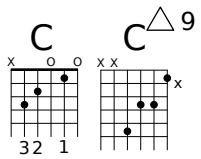


The predefined fret diagram table for guitar contains eight chords (major, minor, augmented, diminished, dominant seventh, major seventh, minor seventh, dominant ninth) for each of 17 keys. The predefined fret diagram table for ukulele contains these chords plus an additional three

chords (major sixth, suspended second, and suspended fourth). A complete list of the predefined fret diagrams is shown in Sezione A.4 [Predefined fretboard diagrams], pagina 621. If there is no entry in the table for a chord, the FretBoards engraver will calculate a fret-diagram using the automatic fret diagram functionality described in [Automatic fret diagrams], pagina 361.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



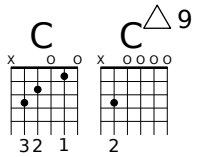
Fret diagrams can be added to the fret diagram table. To add a diagram, you must specify the hash table for the diagram, the chord for the diagram, the tuning to be used, and a definition for the diagram. Normally, the hash table will be *default-fret-table*. The diagram definition can be either a fret-diagram-terse definition string or a fret-diagram-verbose marking list.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c:maj9 }
                        #guitar-tuning
                        #"x;3-2;o;o;o;o;"

mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



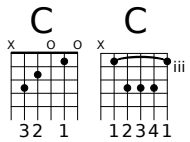
Different fret diagrams for the same chord name can be stored using different octaves of pitches. The different octave should be at least two octaves above or below the default octave, because the octaves above and below the default octave are used for transposing fretboards.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c'' }
                        #guitar-tuning
                        #(offset-fret 2 (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
  c1 c''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



In addition to fret diagrams, LilyPond stores an internal list of chord shapes. The chord shapes are fret diagrams that can be shifted along the neck to different positions to provide different chords. Chord shapes can be added to the internal list and then used to define predefined fret diagrams. Because they can be moved to various positions on the neck, chord shapes will normally not contain any open strings. Like fret diagrams, chord shapes can be entered as either fret-diagram-terse strings or fret-diagram-verbose marking lists.

```
\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

\addChordShape #'powerf #guitar-tuning #"1-1;3-3;3-4;x;x;x;"

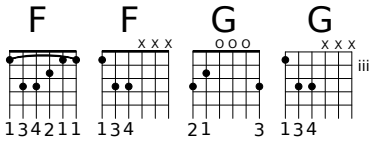
% add some new chords based on the power chord shape

\storePredefinedDiagram #default-fret-table
                        \chordmode { f'' }
                        #guitar-tuning
                        #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
                        \chordmode { g'' }
                        #guitar-tuning
                        #(offset-fret 2 (chord-shape 'powerf guitar-tuning))

mychords = \chordmode{
```

```
  f1 f'' g g''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*. For a predefined fret diagram, the interface properties belong to `FretBoards.FretBoard`.

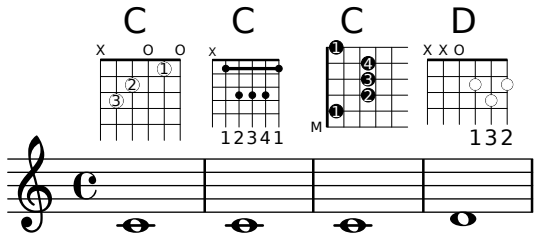
Frammenti di codice selezionati

Customizing fretboard fret diagrams

Fret diagram properties can be set through `'fret-diagram-details`. For FretBoard fret diagrams, overrides are applied to the `FretBoards.FretBoard` object. Like `Voice`, `FretBoards` is a bottom level context, therefore can be omitted in property overrides.

```
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        #"x;1-1-(;3-2;3-3;3-4;1-1-);"
<<
  \new ChordNames {
    \chordmode { c1 | c | c | d }
  }
  \new FretBoards {
    % Set global properties of fret diagram
    \override FretBoards.FretBoard.size = #'1.2
    \override FretBoard.fret-diagram-details.finger-code = #'in-dot
    \override FretBoard.fret-diagram-details.dot-color = #'white
    \chordmode {
      c
      \once \override FretBoard.size = #'1.0
      \once \override FretBoard.fret-diagram-details.barre-type = #'straight
      \once \override FretBoard.fret-diagram-details.dot-color = #'black
      \once \override FretBoard.fret-diagram-details.finger-code = #'below-string
      c'
      \once \override FretBoard.fret-diagram-details.barre-type = #'none
      \once \override FretBoard.fret-diagram-details.number-type = #'arabic
      \once \override FretBoard.fret-diagram-details.orientation = #'landscape
      \once \override FretBoard.fret-diagram-details.mute-string = #"M"
      \once \override FretBoard.fret-diagram-details.label-dir = #LEFT
```

```
      \once \override FretBoard.fret-diagram-details.dot-color = #'black
      c'
      \once \override FretBoard.fret-diagram-details.finger-code = #'below-string
      \once \override FretBoard.fret-diagram-details.dot-radius = #0.35
      \once \override FretBoard.fret-diagram-details.dot-position = #0.5
      \once \override FretBoard.fret-diagram-details.fret-count = #3
      d
    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>
```



Defining predefined fretboards for other instruments

Predefined fret diagrams can be added for new instruments in addition to the standards used for guitar. This file shows how this is done by defining a new string-tuning and a few predefined fretboards for the Venezuelan cuatro.

This file also shows how fingerings can be included in the chords used as reference points for the chord lookup, and displayed in the fret diagram and the `TabStaff`, but not the music.

These fretboards are not transposable because they contain string information. This is planned to be corrected in the future.

```
% add FretBoards for the Cuatro
%   Note: This section could be put into a separate file
%      predefined-cuatro-fretboards.ly
%      and \included into each of your compositions

cuatroTuning = #`(,(ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))

dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }

\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        #"o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        #"o;o;o;3-3;"
```

```
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        #"o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        #"o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                        #cuatroTuning
                        #"2-2;o;1-1;o;"

% end of potential include file /predefined-cuatro-fretboards.ly


#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

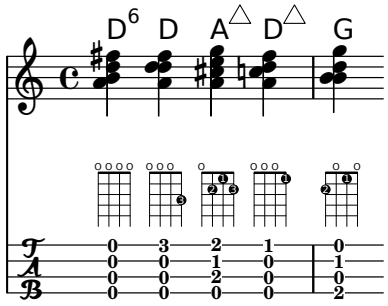
    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
%      \override FretBoard
%        #'(fret-diagram-details string-count) = #'4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }
```

```
  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
    }
  }
  \midi { }
}
```



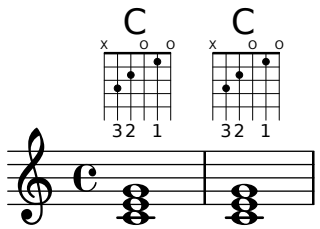
ChordChanges for FretBoards

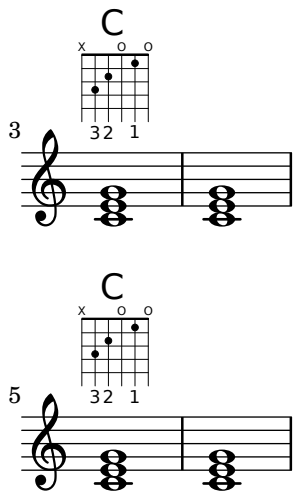
FretBoards can be set to display only when the chord changes or at the beginning of a new line.

```
\include "predefined-guitar-fretboards.ly"

myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
```





Fretboards alternate tables

Alternate fretboard tables can be created. These would be used in order to have alternate fretboards for a given chord.

In order to use an alternate fretboard table, the table must first be created. Fretboards are then added to the table.

The created fretboard table can be blank, or it can be copied from an existing table.

The table to be used in displaying predefined fretboards is selected by the property `\predefinedDiagramTable`.

```
\include "predefined-guitar-fretboards.ly"

% Make a blank new fretboard table
#(define custom-fretboard-table-one (make-fretboard-table))

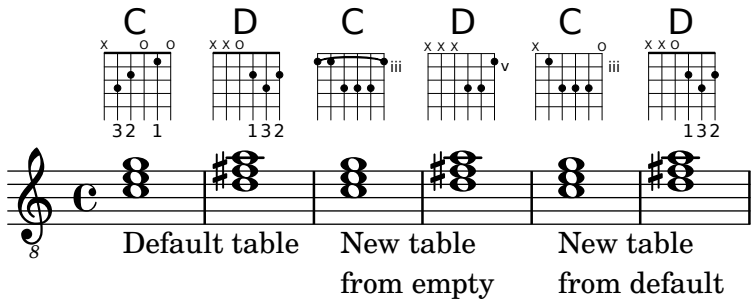
% Make a new fretboard table as a copy of default-fret-table
#(define custom-fretboard-table-two (make-fretboard-table default-fret-table))

% Add a chord to custom-fretboard-table-one
\storePredefinedDiagram #custom-fretboard-table-one
                        \chordmode{c}
                        #guitar-tuning
                        "3-(;3;5;5;5;3-);"

% Add a chord to custom-fretboard-table-two
\storePredefinedDiagram #custom-fretboard-table-two
                        \chordmode{c}
                        #guitar-tuning
                        "x;3;5;5;5;o;"

<<
  \chords {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
  \new FretBoards {
    \chordmode {
      \set predefinedDiagramTable = #default-fret-table
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-one
```

```
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-two
      c1 | d1 |
    }
  }
  \new Staff {
    \clef "treble_8"
    <<
      \chordmode {
        c1 | d1 |
        c1 | d1 |
        c1 | d1 |
      }
      {
        s1_\markup "Default table" | s1 |
        s1_\markup \column {"New table" "from empty"} | s1 |
        s1_\markup \column {"New table" "from default"} | s1 |
      }
    >>
  }
>>
```



Vedi anche

Notation Reference: [Custom tablatures], pagina 339, [Automatic fret diagrams], pagina 361, [Chord mode overview], pagina 392, Sezione A.4 [Predefined fretboard diagrams], pagina 621.

Installed Files: ‘`ly/predefined-guitar-fretboards.ly`’, ‘`ly/predefined-guitar-ninth-fretboards.ly`’, ‘`ly/predefined-ukulele-fretboards.ly`’, ‘`ly/predefined-mandolin-fretboards.ly`’.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

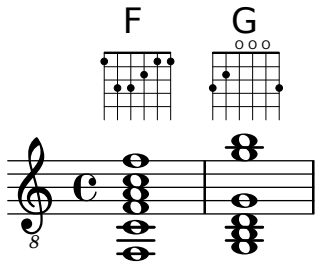
Internals Reference: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

Automatic fret diagrams

Fret diagrams can be automatically created from entered notes using the `FretBoards` context. If no predefined diagram is available for the entered notes in the active `stringTunings`, this context calculates strings and frets that can be used to play the notes.

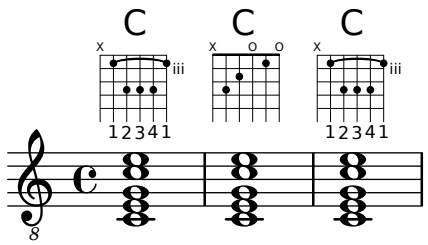
```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
```

```
  }
  \new FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1
    <g, b, d g b' g'>1
  }
>>
```



As no predefined diagrams are loaded by default, automatic calculation of fret diagrams is the default behavior. Once default diagrams are loaded, automatic calculation can be enabled and disabled with predefined commands:

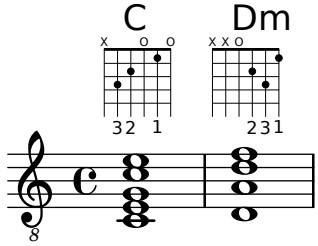
```
\storePredefinedDiagram #default-fret-table
                        <c e g c' e'>
                        #guitar-tuning
                        #"x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
  \new ChordNames {
    \chordmode {
      c1 c c
    }
  }
  \new FretBoards {
    <c e g c' e'>1
    \predefinedFretboardsOff
    <c e g c' e'>1
    \predefinedFretboardsOn
    <c e g c' e'>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <c e g c' e'>1
    <c e g c' e'>1
  }
>>
```



Sometimes the fretboard calculator will be unable to find an acceptable diagram. This can often be remedied by manually assigning a note to a string. In many cases, only one note need be manually placed on a string; the rest of the notes will then be placed appropriately by the `FretBoards` context.

Fingerings can be added to FretBoard fret diagrams.

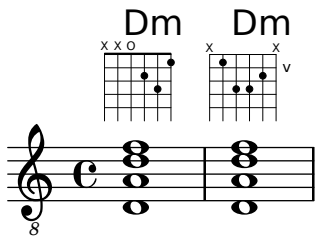
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new FretBoards {
    <c-3 e-2 g c'-1 e'>1
    <d a-2 d'-3 f'-1>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <d a d' f'>1
  }
>>
```



The minimum fret to be used in calculating strings and frets for the FretBoard context can be set with the `minimumFret` property.

```
<<
  \new ChordNames {
    \chordmode {
      d1:m d:m
    }
  }
  \new FretBoards {
    <d a d' f'>1
    \set FretBoards.minimumFret = #5
    <d a d' f'>1
  }
  \new Staff {
    \clef "treble_8"
    <d a d' f'>1
```

```
    <d a d' f'>1
  }
>>
```



The strings and frets for the `FretBoards` context depend on the `stringTunings` property, which has the same meaning as in the TabStaff context. See [Custom tablatures], pagina 339 for information on the `stringTunings` property.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*. For a `FretBoards` fret diagram, the interface properties belong to `FretBoards.FretBoard`.

Comandi predefiniti

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

Vedi anche

Notation Reference: [Custom tablatures], pagina 339.

Snippets: Sezione "Fretted strings" in *Frammenti di codice*.

Internals Reference: Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti
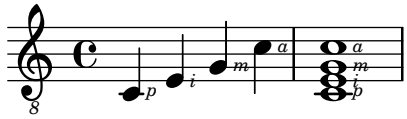
Automatic fretboard calculations do not work properly for instruments with non-monotonic tunings.

Right-hand fingerings

Right-hand fingerings *p-i-m-a* must be entered using `\rightHandFinger` followed by a number.

> **Nota:** If the number is entered in Scheme notation, remember to append a space before following it with a closing `>` or similar.

```
\clef "treble_8"
c4\rightHandFinger #1
e\rightHandFinger #2
g\rightHandFinger #3
c\rightHandFinger #4
<c,\rightHandFinger #1 e\rightHandFinger #2
 g\rightHandFinger #3 c\rightHandFinger #4 >1
```



For convenience, you can abbreviate `\rightHandFinger` to something short, for example `RH`,

```
RH=#rightHandFinger
```

Frammenti di codice selezionati

Placement of right-hand fingerings

It is possible to exercise greater control over the placement of right-hand fingerings by setting a specific property, as demonstrated in the following example. Note: you must use a chord construct

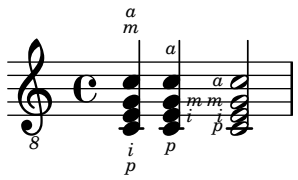
```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"

  \set strokeFingerOrientations = #'(up down)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >4

  \set strokeFingerOrientations = #'(up right down)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >4

  \set strokeFingerOrientations = #'(left)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >2
}
```

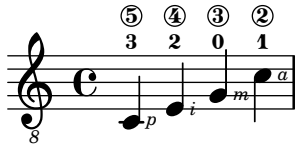


Fingerings string indications and right-hand fingerings

This example combines left-hand fingering, string indications, and right-hand fingering.

```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  <c-3\5-\RH #1 >4
  <e-2\4-\RH #2 >4
  <g-0\3-\RH #3 >4
  <c-1\2-\RH #4 >4
}
```



Vedi anche

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “StrokeFinger” in *Guida al Funzionamento Interno*.
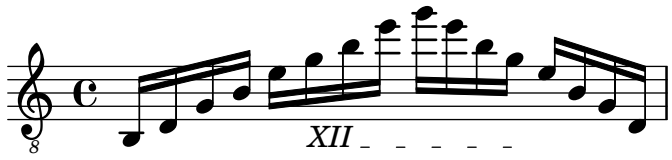
2.4.2 Guitar

Most of the notational issues associated with guitar music are covered sufficiently in the general fretted strings section, but there are a few more worth covering here. Occasionally users want to create songbook-type documents having only lyrics with chord indications above them. Since LilyPond is a music typesetter, it is not recommended for documents that have no music notation in them. A better alternative is a word processor, text editor, or, for experienced users, a typesetter like GuitarTeX.

Indicating position and barring

This example demonstrates how to include guitar position and barring indications.

```
\clef "treble_8"
b16 d g b e
\textSpannerDown
\override TextSpanner.bound-details.left.text = #"XII "
g16\startTextSpan
b16 e g e b g\stopTextSpan
e16 b g d
```



Vedi anche
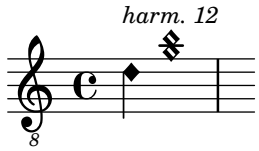
Notation Reference: ⟨undefined⟩ [Text spanners], pagina ⟨undefined⟩.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*, Sezione “Expressive marks” in *Frammenti di codice*.

Indicating harmonics and dampened notes

Special note heads can be used to indicate dampened notes or harmonics. Harmonics are normally further explained with a text markup.

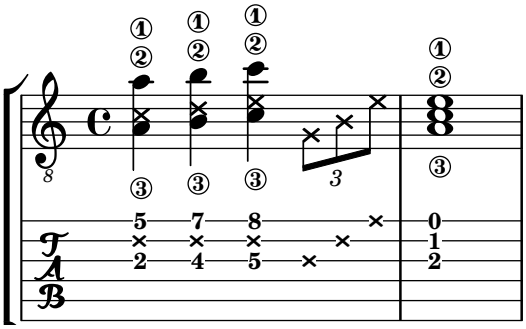
```
\relative c' {
  \clef "treble_8"
  \override Staff.NoteHead.style = #'harmonic-mixed
  d^\markup { \italic { \fontsize #-2 { "harm. 12" }}} <g b>1
}
```



Dampened notes (also called *dead notes*) are supported within normal and tablature staves:

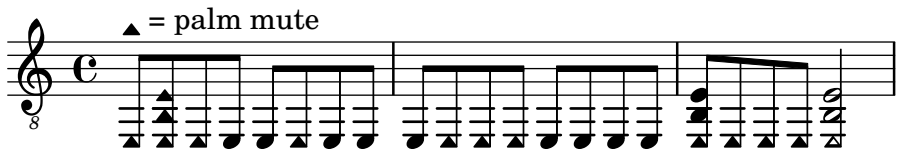
```
music = \relative c' {
  < a\3 \deadNote c\2 a'\1 >4
  < b\3 \deadNote d\2 b'\1 >
  < c\3 \deadNote e\2 c'\1 >
  \deadNotesOn
  \tuplet 3/2 { g8 b e }
```

```
  \deadNotesOff
  < a,\3 c\2 e\1 >1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \music
  }
  \new TabStaff {
    \music
  }
>>
```



Another playing technique (especially used on electric guitars) is called *palm mute*. The string is hereby partly muted by the palm of the striking hand (hence the name). Lilypond supports the notation of palm mute-style notes by changing the note head to a triangle shape.

```
\new Voice { % Warning: explicit Voice instantiation is
             %    required to have palmMuteOff work properly
             %    when palmMuteOn comes at the beginning of
             %    the piece.
  \relative c, {
    \clef "G_8"
    \palmMuteOn
    e8^\markup { \musicglyph #"noteheads.u2do"  = palm mute }
    < e b' e > e
    \palmMuteOff
    e e  \palmMute e e e |
    e8 \palmMute { e e e } e e e e |
    < \palmMute e b' e >8 \palmMute { e e e } < \palmMute e b' e >2
  }
}
```



Vedi anche

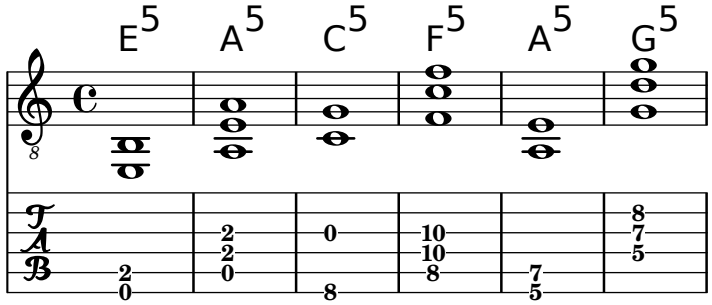
Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Notation Reference: ⟨undefined⟩ [Special note heads], pagina ⟨undefined⟩, Sezione A.9 [Note head styles], pagina 653.

Indicating power chords

Power chords and their symbols can be engraved in chord mode or as chord constructs:

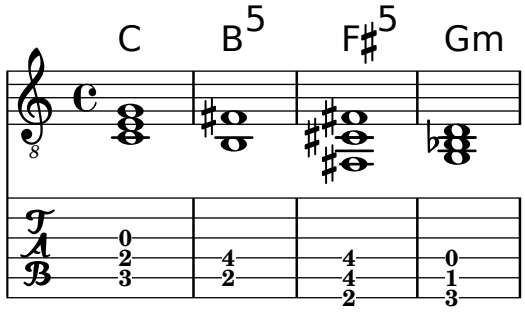
```
ChordsAndSymbols = {
  \chordmode {
    \powerChords
    e,,1:1.5
    a,,1:1.5.8
    \set minimumFret = #8
    c,1:1.5
    f,1:1.5.8
  }
  \set minimumFret = #5
  <a, e>1
  <g d' g'>1
}
\score {
  <<
    \new ChordNames {
    \ChordsAndSymbols
    }
    \new Staff {
      \clef "treble_8"
      \ChordsAndSymbols
    }
    \new TabStaff {
      \ChordsAndSymbols
    }
  >>
}
```



Power chord symbols are automatically switched off as soon as one of the other common chord modifier is used:

```
mixedChords = \chordmode {
  c,1
  \powerChords
  b,,1:1.5
  fis,,1:1.5.8
  g,,1:m
}
\score {
  <<
    \new ChordNames {
```

```
      \mixedChords
    }
    \new Staff {
      \clef "treble_8"
      \mixedChords
    }
    \new TabStaff {
      \mixedChords
    }
  >>
}
```



Vedi anche

Music Glossary: Sezione “power chord” in *Glossario Musicale*.

Notation Reference: [Extended and altered chords], pagina 395, [Printing chord names], pagina 398.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

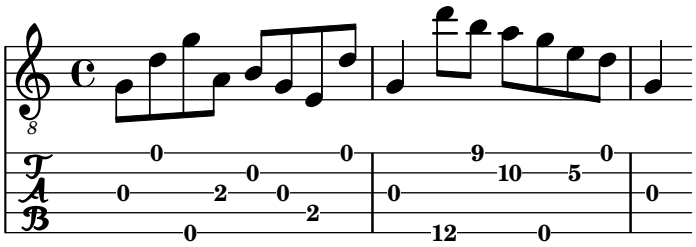
2.4.3 Banjo

Banjo tablatures

LilyPond has basic support for the five-string banjo. When making tablatures for five-string banjo, use the banjo tablature format function to get correct fret numbers for the fifth string:

```
music = {
  g8 d' g'\5 a b g e d' |
  g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
  g4
}

<<
  \new Staff \with { \omit StringNumber }
  { \clef "treble_8"  \music }
  \new TabStaff \with {
    tablatureFormat = #fret-number-tablature-format-banjo
    stringTunings = #banjo-open-g-tuning
  }
  { \music }
>>
```



A number of common tunings for the five-string banjo are predefined: `banjo-c-tuning` (gCGBD), `banjo-modal-tuning` (gDGCD), `banjo-open-d-tuning` (aDF#AD) and `banjo-open-dm-tuning` (aDFAD).

These may be converted to four-string tunings using the `four-string-banjo` function:

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

Vedi anche

Installed Files: '`ly/string-tunings-init.ly`'.

Snippets: Sezione "Fretted strings" in *Frammenti di codice*.

2.5 Percussion

2.5.1 Common notation for percussion

Rhythmic music is primarily used for percussion and drum notation, but it can also be used to show the rhythms of melodies.

References for percussion

- Some percussion may be notated on a rhythmic staff; this is discussed in ⟨undefined⟩ [Showing melody rhythms], pagina ⟨undefined⟩, and ⟨undefined⟩ [Instantiating new staves], pagina ⟨undefined⟩.
- MIDI output is discussed in a separate section; please see Sezione 3.5.6 [Percussion in MIDI], pagina 499.

Vedi anche

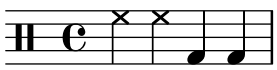
Notation Reference: ⟨undefined⟩ [Showing melody rhythms], pagina ⟨undefined⟩, ⟨undefined⟩ [Instantiating new staves], pagina ⟨undefined⟩. Sezione 3.5.6 [Percussion in MIDI], pagina 499.

Snippets: Sezione "Percussion" in *Frammenti di codice*.

Basic percussion notation

Percussion notes may be entered in `\drummode` mode, which is similar to the standard mode for entering notes. The simplest way to enter percussion notes is to use the `\drums` command, which creates the correct context and entry mode for percussion:

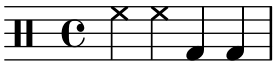
```
\drums {
  hihat4 hh bassdrum bd
}
```



This is shorthand for:

```
\new DrumStaff {
  \drummode {
    hihat4 hh bassdrum bd
  }
```
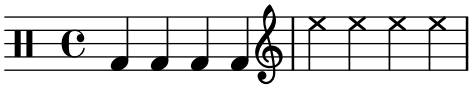
```
}
```



Each piece of percussion has a full name and an abbreviated name, and both can be used in input files. The full list of percussion note names may be found in Sezione A.14 [Percussion notes], pagina 707.

Note that the normal notation of pitches (such as `cis4`) in a `DrumStaff` context will cause an error message. Percussion clefs are added automatically to a `DrumStaff` context but they can also be set explicitly. Other clefs may be used as well.

```
\drums {
  \clef percussion
  bd4 bd bd bd
  \clef treble
  hh4 hh hh hh
}
```



There are a few issues concerning MIDI support for percussion instruments; for details please see Sezione 3.5.6 [Percussion in MIDI], pagina 499.

Vedi anche

Notation Reference: Sezione 3.5.6 [Percussion in MIDI], pagina 499, Sezione A.14 [Percussion notes], pagina 707.
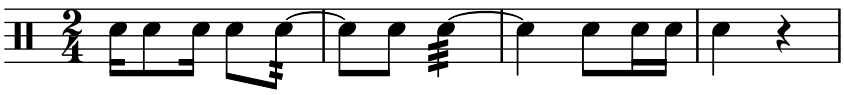
Installed Files: ‘`ly/drumpitch-init.ly`’.

Snippets: Sezione “Percussion” in *Frammenti di codice*.

Drum rolls

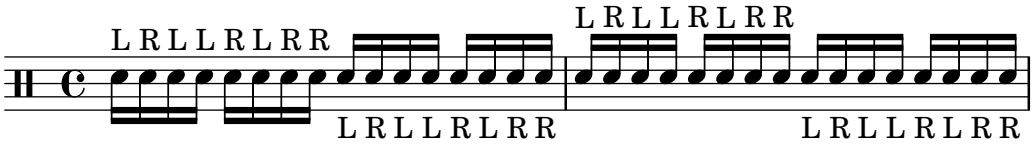
Drum rolls are indicated with three slashes across the stem. For quarter notes or longer the three slashes are shown explicitly, eighth notes are shown with two slashes (the beam being the third), and drum rolls shorter than eighths have one stem slash to supplement the beams. This is achieved with the tremolo notation, as described in ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩.

```
\drums {
  \time 2/4
  sn16 sn8 sn16 sn8 sn8:32 ~
  sn8 sn8 sn4:32 ~
  sn4 sn8 sn16 sn16
  sn4 r4
}
```



Sticking can be indicated by placing a markup for `"R"` or `"L"` above or below notes, as discussed in Sezione 5.4.2 [Direction and placement], pagina 585. The `staff-padding` property may be overridden to achieve a pleasing baseline.

```
\drums {
  \repeat unfold 2 {
    sn16^"L" sn^"R" sn^"L" sn^"L" sn^"R" sn^"L" sn^"R" sn^"R"
    \stemUp
    sn16_"L" sn_"R" sn_"L" sn_"L" sn_"R" sn_"L" sn_"R" sn_"R"
  }
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩.

Snippets: Sezione “Percussion” in *Frammenti di codice*.

Pitched percussion

Certain pitched percussion instruments (e.g. xylophone, vibraphone, and timpani) are written using normal staves. This is covered in other sections of the manual.

Vedi anche

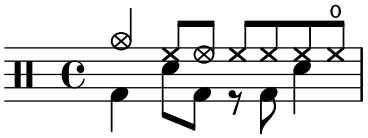
Notation Reference: Sezione 3.5.6 [Percussion in MIDI], pagina 499.

Snippets: Sezione “Percussion” in *Frammenti di codice*.

Percussion staves

A percussion part for more than one instrument typically uses a multiline staff where each position in the staff refers to one piece of percussion. To typeset the music, the notes must be interpreted in `DrumStaff` and `DrumVoice` context.

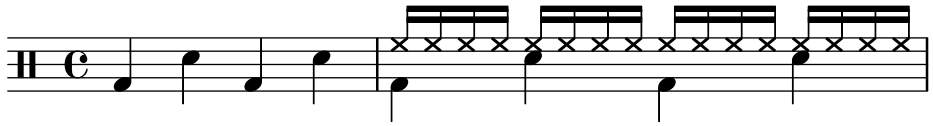
```
up = \drummode {
  crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat
}
down = \drummode {
  bassdrum4 snare8 bd r bd sn4
}
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



The above example shows verbose polyphonic notation. The short polyphonic notation, described in Sezione “I’m hearing Voices” in *Manuale di Apprendimento*, can also be used. For example,

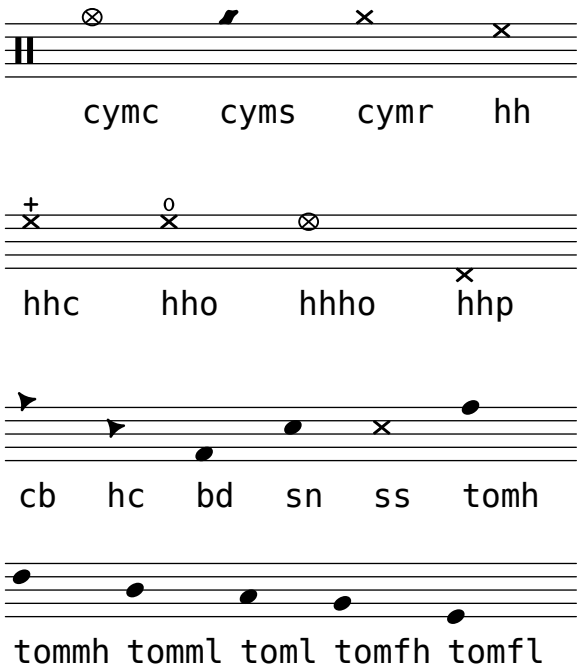
```
\new DrumStaff <<
  \drummode {
```

```
    bd4 sn4 bd4 sn4
    << {
      \repeat unfold 16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    } >>
  }
>>
```



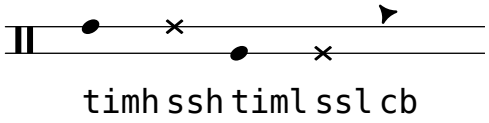
There are also other layout possibilities. To use these, set the property `drumStyleTable` in context `DrumVoice`. The following variables have been predefined:

`drums-style`
: This is the default. It typesets a typical drum kit on a five-line staff:

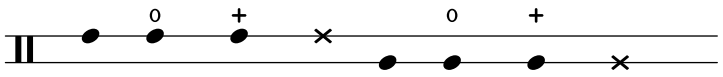


  The drum scheme supports six different toms. When there are fewer toms, simply select the toms that produce the desired result. For example, to get toms on the three middle lines you use `tommh`, `tomml`, and `tomfh`.

`timbales-style`
: This typesets timbales on a two line staff:

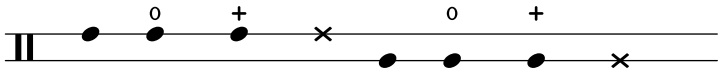


`congas-style`
: This typesets congas on a two line staff:



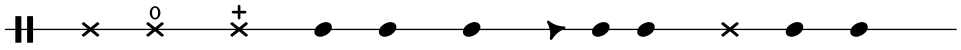
`bongos-style`

This typesets bongos on a two line staff:



`percussion-style`

To typeset all kinds of simple percussion on one line staves:

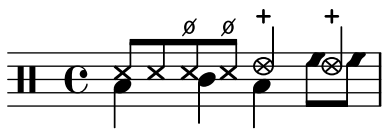


Custom percussion staves

If you do not like any of the predefined lists you can define your own list at the top of your file.

```
#(define mydrums '(
         (bassdrum        default   #f           -1)
         (snare           default   #f           0)
         (hihat           cross     #f           1)
         (halfopenhihat   cross     "halfopen"   1)
         (pedalhihat      xcircle   "stopped"    2)
         (lowtom          diamond   #f           3)))
up = \drummode { hh8 hh hhho hhho hhp4 hhp }
down = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff <<
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



Frammenti di codice selezionati

Here are some examples:

Two Woodblocks, entered with wbh (high woodblock) and wbl (low woodblock)

```
% These lines define the position of the woodblocks in the stave;
% if you like, you can change it or you can use special note heads
% for the woodblocks.
#(define mydrums '((hiwoodblock default #t  3)
                   (lowoodblock default #t -2)))

woodstaff = {
  % This defines a staff with only two lines.
```

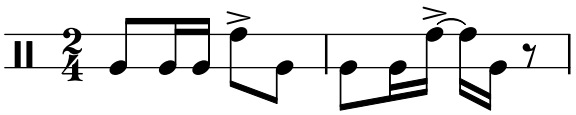
```
  % It also defines the positions of the two lines.
  \override Staff.StaffSymbol.line-positions = #'(-2 3)

  % This is necessary; if not entered, the barline would be too short!
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
}

\new DrumStaff {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  % with this you load your new drum style table
  \woodstaff

  \drummode {
    \time 2/4
    wbl8 wbl16 wbl wbh8-> wbl |
    wbl8 wbl16 wbh-> ~ wbh wbl16 r8 |
  }
}
```



Note that in this special case the length of the barline must altered with `\override Staff.BarLine.bar-extent #'(from . to)`. Otherwise it would be too short. And you have also to define the positions of the two stafflines. For more information about these delicate things have a look at ⟨undefined⟩ [Staff symbol], pagina ⟨undefined⟩.

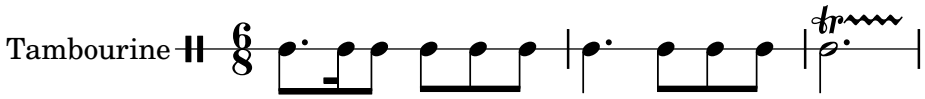
A tambourine, entered with 'tamb':

```
#(define mydrums '((tambourine default #t 0)))

tambustaff = {
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Tambourine"
}

\new DrumStaff {
  \tambustaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    \time 6/8
    tamb8. tamb16 tamb8 tamb tamb tamb |
    tamb4. tamb8 tamb tamb |
    % the trick with the scaled duration and the shorter rest
    % is neccessary for the correct ending of the trill-span!
    tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
  }
}
```



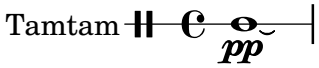
Music for Tam-Tam (entered with ‘tt’):

```
#(define mydrums '((tamtam default #t 0)))

tamtamstaff = {
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Tamtam"
}

\new DrumStaff {
  \tamtamstaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    tt 1 \pp \laissezVibrer
  }
}
```

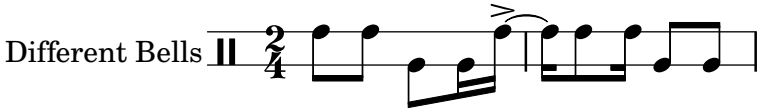


Two different bells, entered with ‘cb’ (cowbell) and ‘rb’ (ridebell)

```
#(define mydrums '((ridebell default #t  3)
                   (cowbell  default #t -2)))

bellstaff = {
  \override DrumStaff.StaffSymbol.line-positions = #'(-2 3)
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Different Bells"
}

\new DrumStaff {
  \bellstaff
  \drummode {
    \time 2/4
    rb8 rb cb cb16 rb-> ~ |
    rb16 rb8 rb16 cb8 cb |
  }
}
```



Here a short example taken from Stravinsky’s ‘L’histoire du Soldat’.

```
#(define mydrums '((bassdrum   default #t  4)
                   (snare      default #t -4)
                   (tambourine default #t  0)))
```

```
global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}
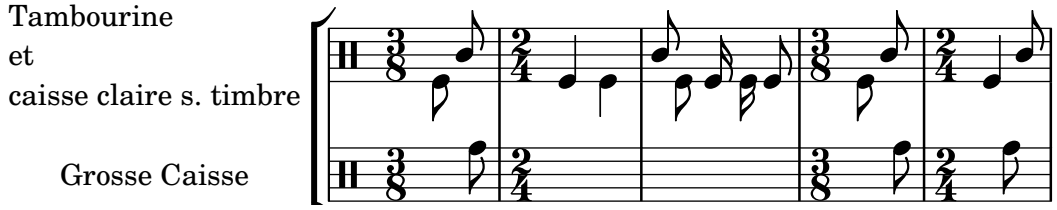
drumsA = {
  \context DrumVoice <<
    { \global }
    { \drummode {
        \autoBeamOff
        \stemDown sn8 \stemUp tamb s8 |
        sn4 \stemDown sn4 |
        \stemUp tamb8 \stemDown sn8 \stemUp sn16 \stemDown sn \stemUp sn8 |
        \stemDown sn8 \stemUp tamb s8 |
        \stemUp sn4 s8 \stemUp tamb
      }
    }
  >>
}

drumsB = {
  \drummode {
    s4 bd8 s2*2 s4 bd8 s4 bd8 s8
  }
}

\layout {
  indent = #40
}

\score {
  \new StaffGroup <<
    \new DrumStaff {
      \set DrumStaff.instrumentName = \markup {
        \column {
          "Tambourine"
          "et"
          "caisse claire s. timbre"
        }
      }
      \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
      \drumsA
    }

   \new DrumStaff {
     \set DrumStaff.instrumentName = #"Grosse Caisse"
     \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
     \drumsB }
  >>
}
```



Vedi anche

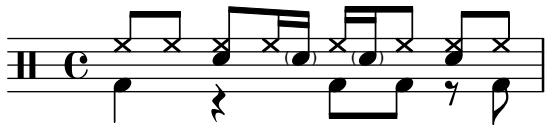
Snippets: Sezione “Percussion” in *Frammenti di codice*.

Internals Reference: Sezione “DrumStaff” in *Guida al Funzionamento Interno*, Sezione “DrumVoice” in *Guida al Funzionamento Interno*.

Ghost notes

Ghost notes for drums and percussion may be created using the `\parenthesize` command detailed in ⟨undefined⟩ [Parentheses], pagina ⟨undefined⟩.
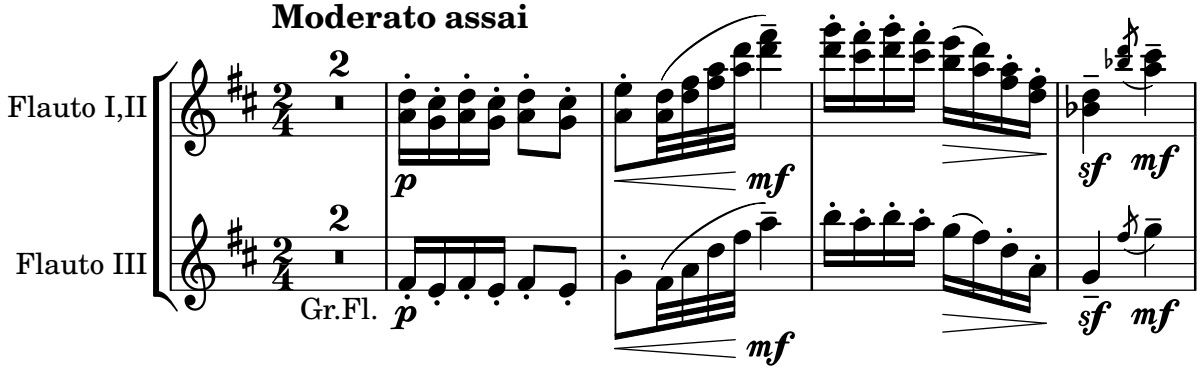
```
\new DrumStaff
<<
  \context DrumVoice  = "1" { s1 }
  \context DrumVoice  = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ hh] <hh sn> hh16
        \parenthesize sn  hh
        \parenthesize sn  hh8 <hh sn> hh
      } \\
      {
        bd4 r4 bd8 bd r8 bd
      }
    >>
  }
>>
```



Vedi anche

Snippets: Sezione “Percussion” in *Frammenti di codice*.

2.6 Wind instruments



This section includes elements of music notation that arise when writing specifically for wind instruments.

2.6.1 Common notation for wind instruments

This section discusses notation common to most wind instruments.

References for wind instruments

Many notation issues for wind instruments pertain to breathing and tonguing:

- Breathing can be specified by rests or ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩.
- Legato playing is indicated by ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩.
- Different types of tonguings, ranging from legato to non-legato to staccato are usually shown by articulation marks, sometimes combined with slurs, see ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩ and Sezione A.13 [List of articulations], pagina 706.
- Flutter tonguing is usually indicated by placing a tremolo mark and a text markup on the note. See ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩.

Other aspects of musical notation that can apply to wind instruments:

- Many wind instruments are transposing instruments, see ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩.
- Slide glissandi are characteristic of the trombone, but other winds may perform keyed or valved glissandi. See [Glissando], pagina 133.
- Harmonic series glissandi, which are possible on all brass instruments but common for French Horns, are usually written out as ⟨undefined⟩ [Grace notes], pagina ⟨undefined⟩.
- Pitch inflections at the end of a note are discussed in ⟨undefined⟩ [Falls and doits], pagina ⟨undefined⟩.
- Key slaps or valve slaps are often shown by the `cross` style of ⟨undefined⟩ [Special note heads], pagina ⟨undefined⟩.
- Woodwinds can overblow low notes to sound harmonics. These are shown by the `flageolet` articulation. See Sezione A.13 [List of articulations], pagina 706.
- The use of brass mutes is usually indicated by a text markup, but where there are many rapid changes it is better to use the `stopped` and `open` articulations. See ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩ and Sezione A.13 [List of articulations], pagina 706.
- Stopped horns are indicated by the `stopped` articulation. See ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.

Frammenti di codice selezionati

Changing \flageolet mark size
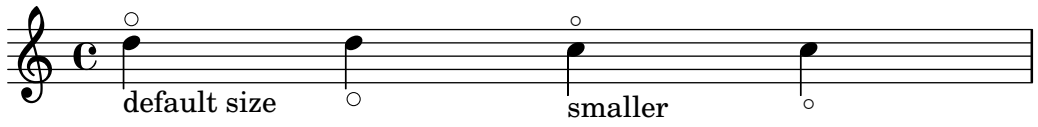
To make the `\flageolet` circle smaller use the following Scheme function.

```
smallFlageolet =
#(let ((m (make-articulation "flageolet")))
   (set! (ly:music-property m 'tweaks)
         (acons 'font-size -3
                (ly:music-property m 'tweaks)))
   m)

\layout { ragged-right = ##f }

\relative c'' {
  d4^\flageolet_\markup { default size } d_\flageolet
```

```
  c4^\smallFlageolet_\markup { smaller } c_\smallFlageolet
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩, ⟨undefined⟩ [Slurs], pagina ⟨undefined⟩, ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩, Sezione A.13 [List of articulations], pagina 706, ⟨undefined⟩ [Tremolo repeats], pagina ⟨undefined⟩, ⟨undefined⟩ [Instrument transpositions], pagina ⟨undefined⟩, [Glissando], pagina 133, ⟨undefined⟩ [Grace notes], pagina ⟨undefined⟩, ⟨undefined⟩ [Falls and doits], pagina ⟨undefined⟩, ⟨undefined⟩ [Special note heads], pagina ⟨undefined⟩,

Snippets: Sezione “Winds” in *Frammenti di codice*.

Fingerings

All wind instruments other than the trombone require the use of several fingers to produce each pitch. Some fingering examples are shown in the snippets below.

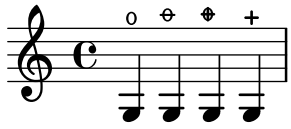
Woodwind diagrams can be produced and are described in Sezione 2.6.3.1 [Woodwind diagrams], pagina 384.

Frammenti di codice selezionati

Fingering symbols for wind instruments

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(ly:make-simple-closure
    `(,+
      ,(ly:make-simple-closure (list
        ly:self-alignment-interface::centered-on-x-parent))
      ,(ly:make-simple-closure (list
        ly:self-alignment-interface::x-aligned-on-self))))
}
\score
{\relative c'
  {
    g\open
    \once \override TextScript.staff-padding = #-1.0 \centermarkup
    g^\markup{\combine \musicglyph #"scripts.open" \musicglyph
    #"scripts.tenuto"}
    \centermarkup g^\markup{\combine \musicglyph #"scripts.open"
    \musicglyph #"scripts.stopped"}
    g\stopped
  }
}
```



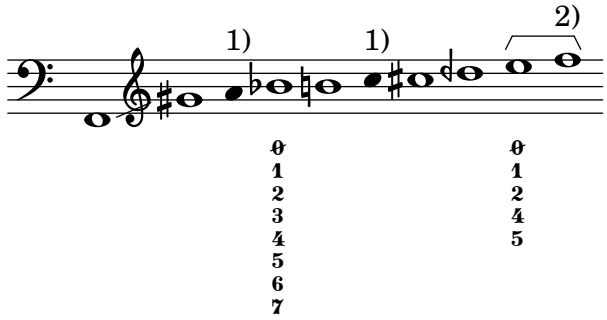
Recorder fingering chart

The following example demonstrates how fingering charts for wind instruments can be realized.

```
% range chart for paetzold contrabass recorder

centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(ly:make-simple-closure
  `(,+
  ,(ly:make-simple-closure (list
  ly:self-alignment-interface::centered-on-x-parent))
  ,(ly:make-simple-closure (list
  ly:self-alignment-interface::x-aligned-on-self))))
}

\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
    \omit Stem
    \omit Flag
    \consists "Horizontal_bracket_engraver"
  }
  {
    \clef bass
    \set Score.timing = ##f
    f,1*1/4 \glissando
    \clef violin
    gis'1*1/4
    \stemDown a'4^\markup{1)}
    \centermarkup
    \once \override TextScript.padding = #2
    bes'1*1/4_\markup{\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 3 \finger 4
    \finger 5 \finger 6 \finger 7} }
    b'1*1/4
    c''4^\markup{1)}
    \centermarkup
    \once \override TextScript.padding = #2
    cis''1*1/4
    deh''1*1/4
    \centermarkup
    \once \override TextScript.padding = #2
    \once \override Staff.HorizontalBracket.direction = #UP
    e''1*1/4_\markup{\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 4
    \finger 5} }\startGroup
    f''1*1/4^\markup{2)}\stopGroup
  }
}
```



Vedi anche

Notation Reference: Sezione 2.6.3.1 [Woodwind diagrams], pagina 384.

Snippets: Sezione “Winds” in *Frammenti di codice*.

2.6.2 Bagpipes

This section discusses notation common bagpipes.

Bagpipe definitions

LilyPond contains special definitions for Scottish, Highland Bagpipe music; to use them, add
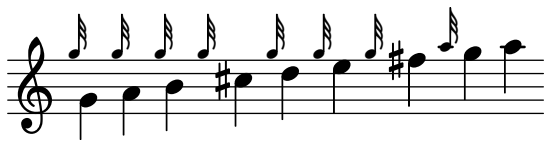
```
\include "bagpipe.ly"
```

to the top of your input file. This lets you add the special grace notes common to bagpipe music with short commands. For example, you could write `\taor` instead of

```
\grace { \small G32[ d G e] }
```

‘`bagpipe.ly`’ also contains pitch definitions for the bagpipe notes in the appropriate octaves, so you do not need to worry about `\relative` or `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



Bagpipe music nominally uses the key of D Major (even though that isn’t really true). However, since that is the only key that can be used, the key signature is normally not written out. To set this up correctly, always start your music with `\hideKeySignature`. If you for some reason want to show the key signature, you can use `\showKeySignature` instead.

Some modern music use cross fingering on c and f to flatten those notes. This can be indicated by `cflat` or `fflat`. Similarly, the piobaireachd high g can be written `gflat` when it occurs in light music.

Vedi anche

Snippets: Sezione “Winds” in *Frammenti di codice*.

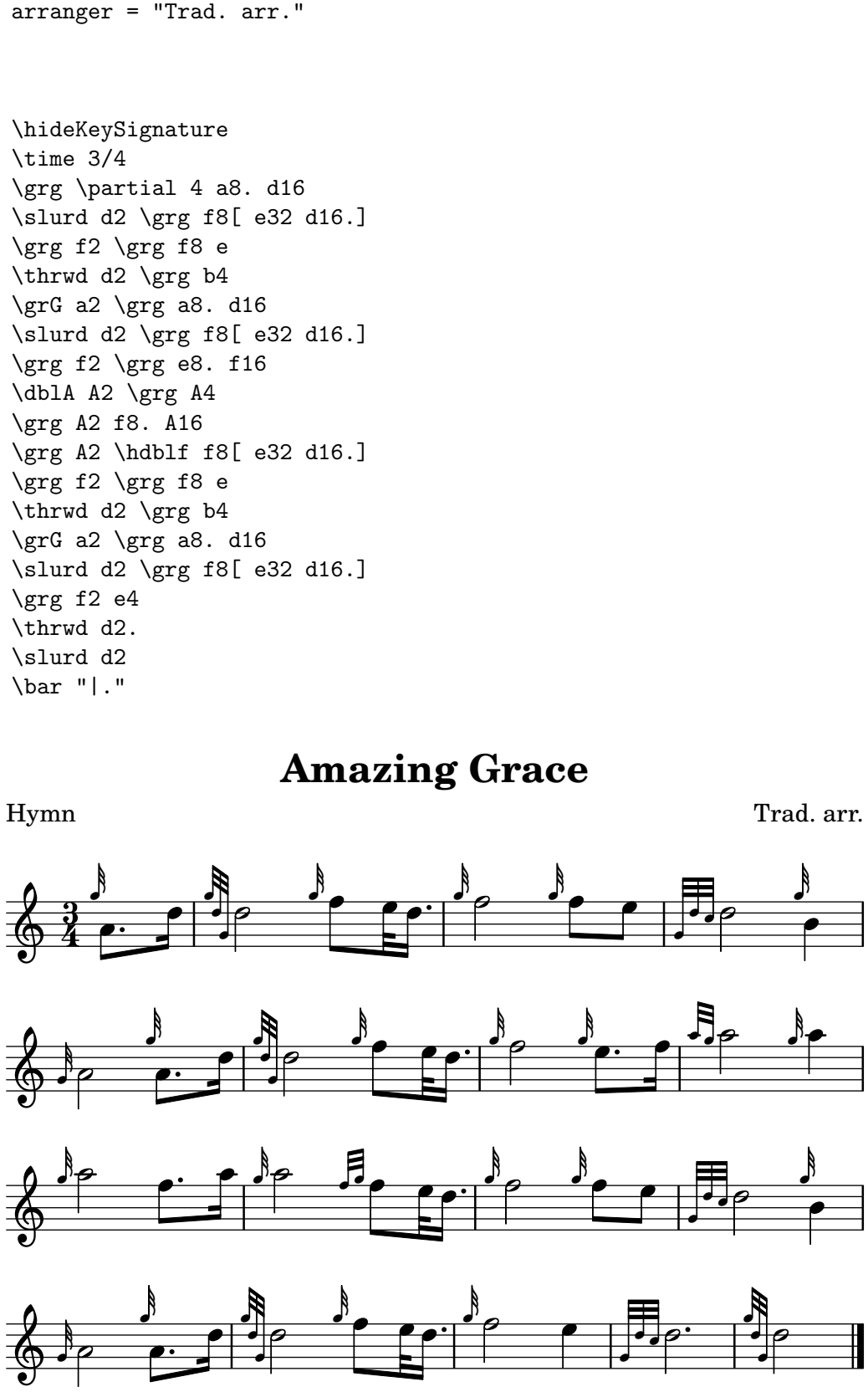
Bagpipe example

This is what the well known tune Amazing Grace looks like in bagpipe notation.

```
\include "bagpipe.ly"
\layout {
  indent = 0.0\cm
  \context { \Score \remove "Bar_number_engraver" }
}
```

```
\header {
  title = "Amazing Grace"
  meter = "Hymn"
  arranger = "Trad. arr."
}

{
  \hideKeySignature
  \time 3/4
  \grg \partial 4 a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg e8. f16
  \dblA A2 \grg A4
  \grg A2 f8. A16
  \grg A2 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
  \thrwd d2.
  \slurd d2
  \bar "|."
}
```



Vedi anche

Snippets: Sezione “Winds” in *Frammenti di codice*.

2.6.3 Woodwinds

This section discusses notation specifically for woodwind instruments.

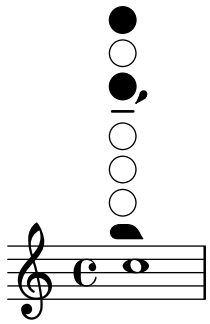
2.6.3.1 Woodwind diagrams

Woodwind diagrams can be used to indicate the fingering to be used for specific notes and are available for the following instruments:

- piccolo
- flute
- oboe
- clarinet
- bass clarinet
- saxophone
- bassoon
- contrabassoon

Woodwind diagrams are created as markups:

```
c1^\markup {
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                 (cc . (one three))
                                 (rh . (ees)))
}
```



Keys can be open, partially-covered, ring-depressed, or fully covered:

```
\textLengthOn
c1^\markup {
  \center-column {
    "one quarter"
    \woodwind-diagram #'flute #'((cc . (one1q))
                                 (lh . ())
                                 (rh . ()))
  }
}

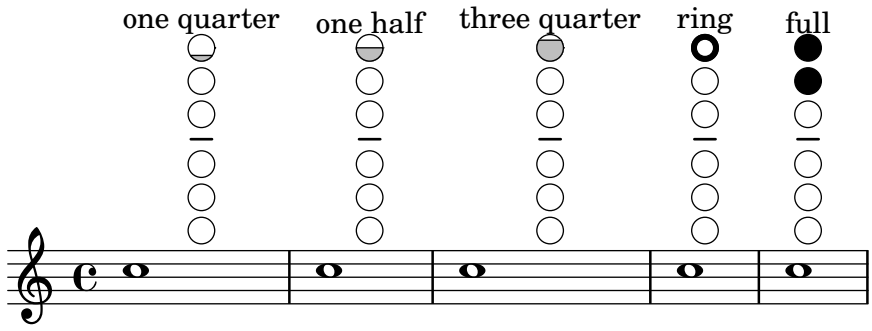
c1^\markup {
  \center-column {
    "one half"
    \woodwind-diagram #'flute #'((cc . (one1h))
```

```
                                (lh . ())
                                (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "three quarter"
    \woodwind-diagram #'flute #'((cc . (one3q))
                                (lh . ())
                                (rh . ()))
  }
}

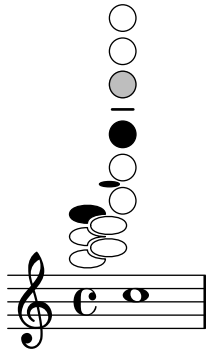
c1^\markup {
  \center-column {
    "ring"
    \woodwind-diagram #'flute #'((cc . (oneR))
                                (lh . ())
                                (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "full"
    \woodwind-diagram #'flute #'((cc . (oneF two))
                                (lh . ())
                                (rh . ()))
  }
}
```



Trills are indicated as shaded keys:

```
c1^\markup {
  \woodwind-diagram #'bass-clarinet
                    #'((cc . (threeT four))
                       (lh . ())
                       (rh . (b fis)))
}
```



A variety of trills can be displayed:

```
\textLengthOn
c1^\markup {
  \center-column {
    "one quarter to ring"
    \woodwind-diagram #'flute #'((cc . (one1qTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

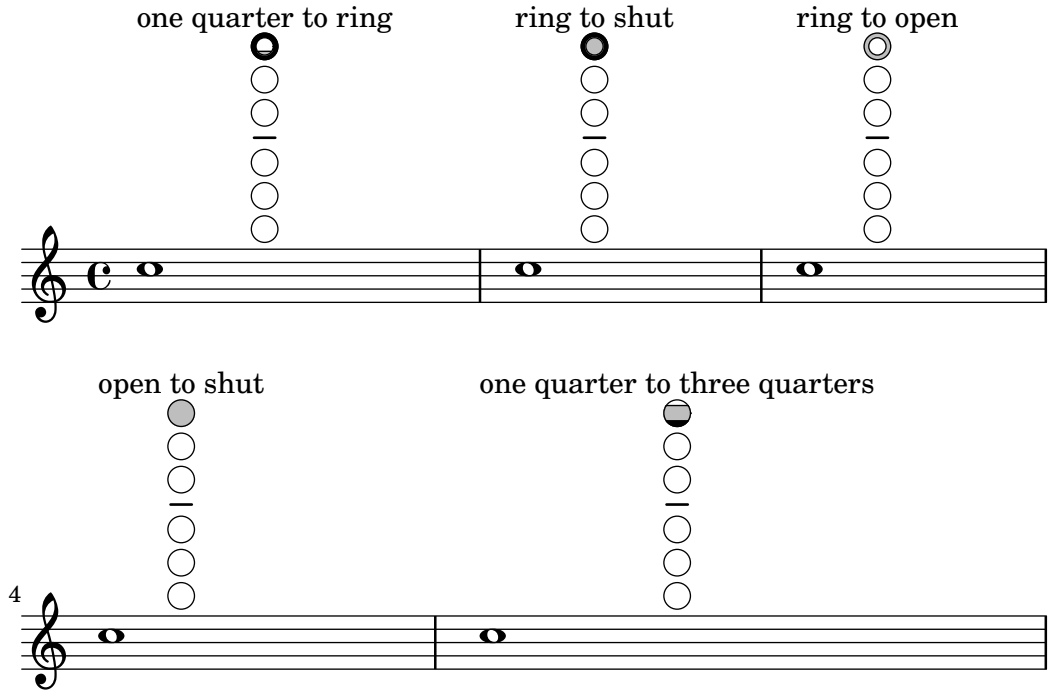
c1^\markup {
  \center-column {
    "ring to shut"
    \woodwind-diagram #'flute #'((cc . (oneTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "ring to open"
    \woodwind-diagram #'flute #'((cc . (oneRT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "open to shut"
    \woodwind-diagram #'flute #'((cc . (oneT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "one quarter to three quarters"
    \woodwind-diagram #'flute #'((cc . (one1qT3q))
                                 (lh . ())
```

```
                                (rh . ()))
  }
}
```



The list of all possible keys and settings for a given instrument can be displayed on the console using `#(print-keys-verbose 'flute)` or in the log file using `#(print-keys-verbose 'flute (current-error-port))`, although they will not show up in the music output.

Creating new diagrams is possible, although this will require Scheme ability and may not be accessible to all users. The patterns for the diagrams are in '`scm/define-woodwind-diagrams.scm`' and '`scm/display-woodwind-diagrams.scm`'.

Comandi predefiniti

Frammenti di codice selezionati

Woodwind diagrams listing

The following music shows all of the woodwind diagrams currently defined in LilyPond.

```
\relative c' {
  \textLengthOn
  c1^
  \markup {
    \center-column {
      'tin-whistle
      " "
       \woodwind-diagram
                 #'tin-whistle
                 #'()
    }
  }

  c1^
  \markup {
```

```
  \center-column {
    'piccolo
    " "
     \woodwind-diagram
                #'piccolo
                #'()
  }
}

c1^
\markup {
  \center-column {
     'flute
     " "
     \woodwind-diagram
        #'flute
        #'()
  }
}
c1^\markup {
  \center-column {
    'oboe
    " "
    \woodwind-diagram
      #'oboe
      #'()
  }
}

c1^\markup {
  \center-column {
    'clarinet
    " "
    \woodwind-diagram
      #'clarinet
      #'()
  }
}

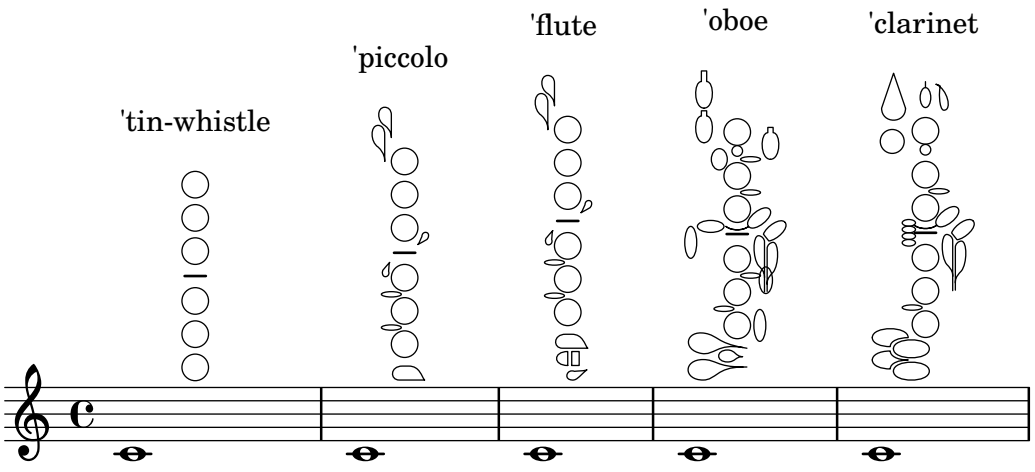
c1^\markup {
  \center-column {
    'bass-clarinet
    " "
    \woodwind-diagram
      #'bass-clarinet
      #'()
  }
}

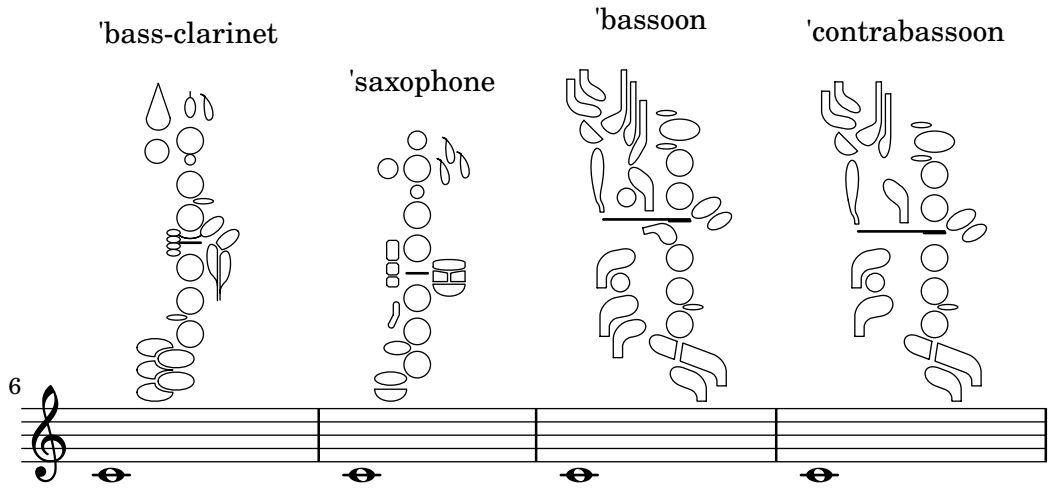
c1^\markup {
  \center-column {
    'saxophone
```

```
      " "
      \woodwind-diagram
        #'saxophone
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'bassoon
      " "
      \woodwind-diagram
        #'bassoon
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'contrabassoon
      " "
      \woodwind-diagram
        #'contrabassoon
        #'()
    }
  }
}
```



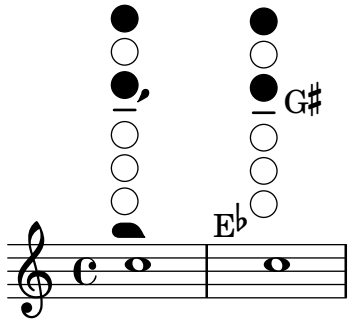


Graphical and text woodwind diagrams

In many cases, the keys other than the central column can be displayed by key name as well as by graphical means.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
      #'((cc . (one three))
         (lh . (gis))
         (rh . (ees)))

  c^\markup
    \override #'(graphical . #f) {
      \woodwind-diagram
        #'piccolo
        #'((cc . (one three))
           (lh . (gis))
           (rh . (ees)))
    }
}
```



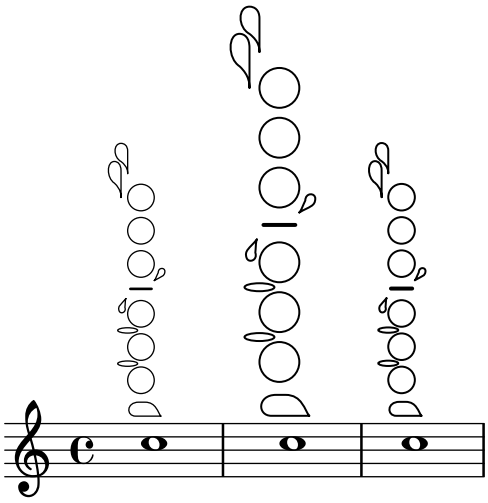
Changing the size of woodwind diagrams

The size and thickness of woodwind diagrams can be changed.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
```

```
      #'()

  c^\markup
    \override #'(size . 1.5) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
  c^\markup
    \override #'(thickness . 0.15) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
}
```



Woodwind diagrams key lists

The snippet below produces a list of all possible keys and key settings for woodwind diagrams as defined in '`scm/define-woodwind-diagrams.scm`'. The list will be displayed in the log file, but not in the music. If output to the console is wanted, omit the `(current-error-port)` from the commands.

```
#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))
```

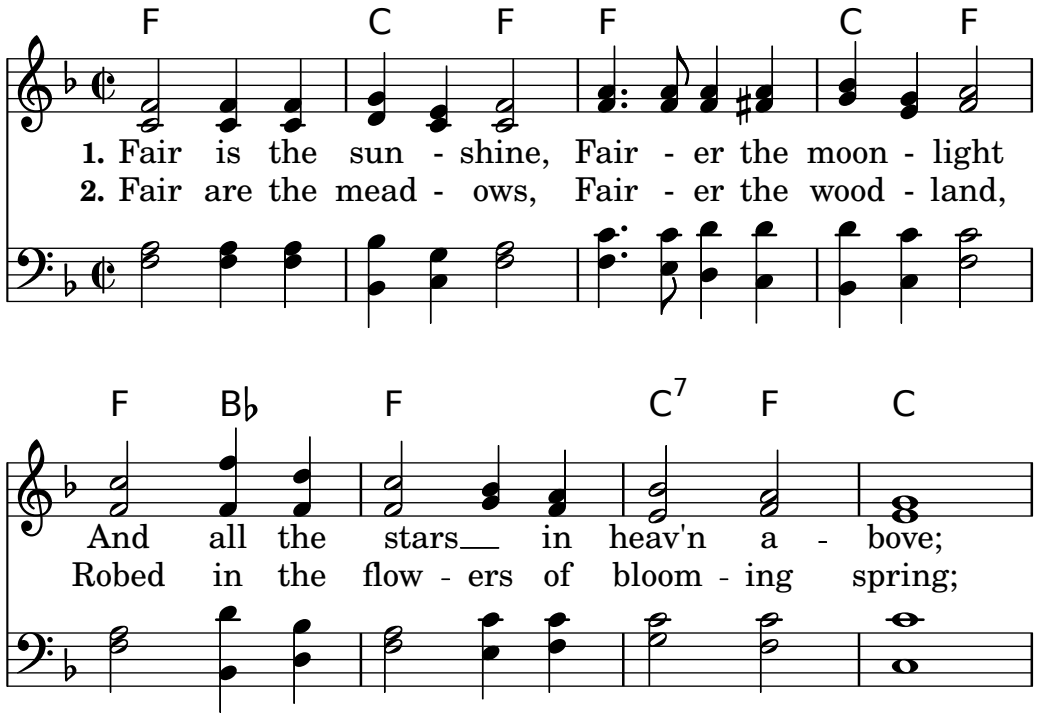
Vedi anche

Installed Files: ‘`scm/define-woodwind-diagrams.scm`’, ‘`scm/display-woodwind-diagrams.scm`’.

Snippets: Sezione “Winds” in *Frammenti di codice*.

Internals Reference: Sezione “TextScript” in *Guida al Funzionamento Interno*, Sezione “instrument-specific-markup-interface” in *Guida al Funzionamento Interno*.

2.7 Chord notation



Chords can be entered either as normal notes or in chord mode and displayed using a variety of traditional European chord naming conventions. Chord names and figured bass notation can also be displayed.

2.7.1 Chord mode
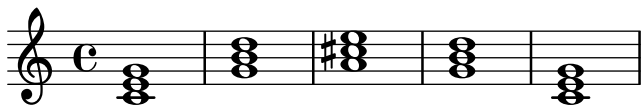
Chord mode is used to enter chords using an indicator of the chord structure, rather than the chord pitches.

Chord mode overview

Chords can be entered as simultaneous music, as discussed in ⟨undefined⟩ [Chorded notes], pagina ⟨undefined⟩.

Chords can also be entered in “chord mode”, which is an input mode that focuses on the structures of chords in traditional European music, rather than on specific pitches. This is convenient for those who are familiar with using chord names to describe chords. More information on different input modes can be found at Sezione 5.4.1 [Input modes], pagina 583.
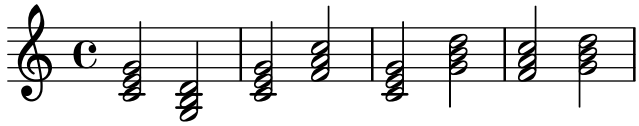
```
\chordmode { c1 g a g c }
```



Chords entered using chord mode are music elements, and can be transposed just like chords entered using simultaneous music. `\chordmode` is absolute, as `\relative` has no effect on `chordmode` blocks. However, in `\chordmode` the absolute pitches are one octave higher than in note mode.

Chord mode and note mode can be mixed in sequential music:

```
<c e g>2 <g b d>
\chordmode { c2 f }
<c e g>2 <g' b d>
\chordmode { f2 g }
```



Vedi anche

Music Glossary: Sezione “chord” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Chorded notes], pagina ⟨undefined⟩, Sezione 5.4.1 [Input modes], pagina 583.
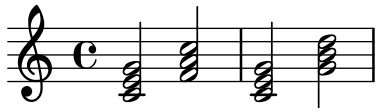
Snippets: Sezione “Chords” in *Frammenti di codice*.

Problemi noti e avvertimenti

Predefined shorthands for articulations and ornaments cannot be used on notes in chord mode, see ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.
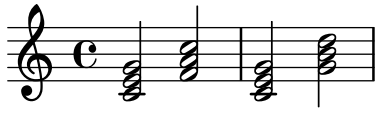
When chord mode and note mode are mixed in sequential music, and chord mode comes first, the note mode will create a new `Staff` context:

```
\chordmode { c2 f }
<c e g>2 <g' b d>
```



To avoid this behavior, explicitly create the `Staff` context:

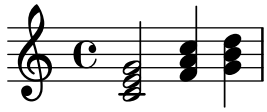
```
\new Staff {
  \chordmode { c2 f }
  <c e g>2  <g' b d>
}
```



Common chords
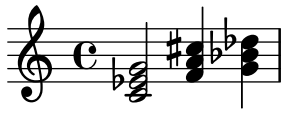
Major triads are entered by including the root and an optional duration:

```
\chordmode { c2 f4 g }
```



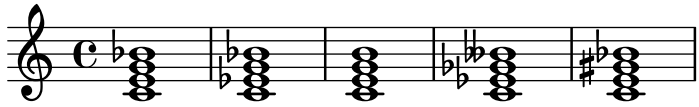
Minor, augmented, and diminished triads are entered by placing `:` and a quality modifier string after the duration:

```
\chordmode { c2:m f4:aug g:dim }
```

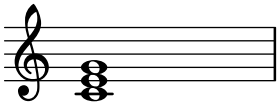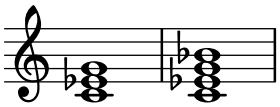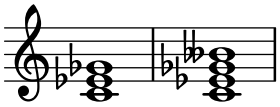
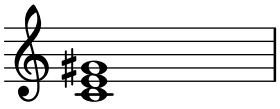
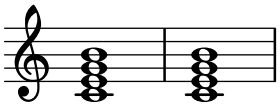


Seventh chords can be created:

```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



The table below shows the actions of the quality modifiers on triads and seventh chords. The default seventh step added to chords is a minor or flatted seventh, which makes the dominant seventh the basic seventh chord. All alterations are relative to the dominant seventh. A more complete table of modifier usage is found at Sezione A.2 [Common chord modifiers], pagina 616.

Modifier	**Action**	**Example**
None	The default action; produces a major triad.	
m, m7	The minor chord. This modifier lowers the 3rd.	
dim, dim7	The diminished chord. This modifier lowers the 3rd, 5th and (if present) the 7th step.	
aug	The augmented chord. This modifier raises the 5th step.	
maj, maj7	The major 7th chord. This modifier adds a raised 7th step. The `7` following `maj` is optional. Do NOT use this modifier to create a major triad.	

Vedi anche

Notation Reference: Sezione A.2 [Common chord modifiers], pagina 616, [Extended and altered chords], pagina 395.

Snippets: Sezione “Chords” in *Frammenti di codice*.

Problemi noti e avvertimenti

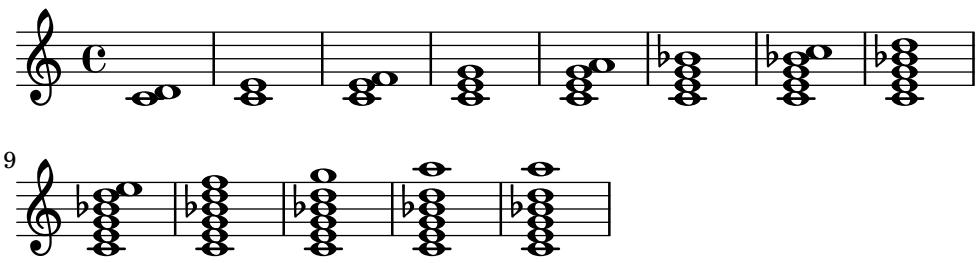
Only one quality modifier should be used per chord, typically on the highest step present in the chord. Chords with more than quality modifier will be parsed without an error or warning, but the results are unpredictable. Chords that cannot be achieved with a single quality modifier should be altered by individual pitches, as described in [Extended and altered chords], pagina 395.

Extended and altered chords

Chord structures of arbitrary complexity can be created in chord mode. The modifier string can be used to extend a chord, add or remove chord steps, raise or lower chord steps, and add a bass note or create an inversion.

The first number following the `:` is taken to be the extent of the chord. The chord is constructed by sequentially adding thirds to the root until the specified number has been reached. Note that the seventh step added as part of an extended chord will be the minor or flatted seventh, not the major seventh. If the extent is not a third (e.g., 6), thirds are added up to the highest third below the extent, and then the step of the extent is added. The largest possible value for the extent is 13. Any larger value is interpreted as 13.

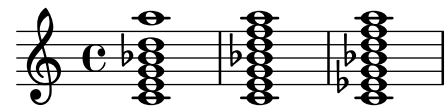
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



Note that both `c:5` and `c` produce a C major triad.

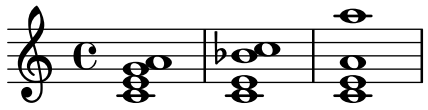
Since an unaltered 11 does not sound good when combined with an unaltered 13, the 11 is removed from a `:13` chord (unless it is added explicitly).

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



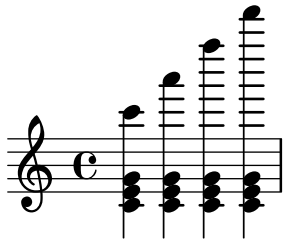
Individual steps can be added to a chord. Additions follow the extent and are prefixed by a dot (`.`). The basic seventh step added to a chord is the minor or flatted seventh, rather than the major seventh.

```
\chordmode {
  c1:5.6 c:3.7.8 c:3.6.13
}
```



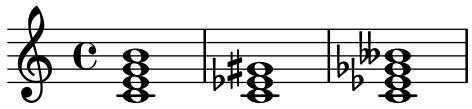
Added steps can be as high as desired.

```
\chordmode {
  c4:5.15 c:5.20 c:5.25 c:5.30
}
```



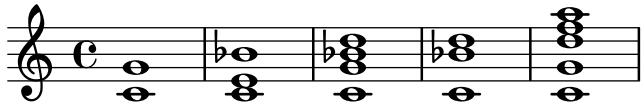
Added chord steps can be altered by suffixing a `-` or `+` sign to the number. To alter a step that is automatically included as part of the basic chord structure, add it as an altered step.

```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```
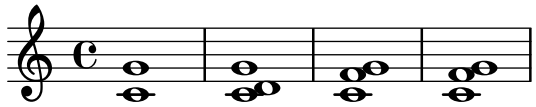


Following any steps to be added, a series of steps to be removed is introduced in a modifier string with a prefix of `^`. If more than one step is to be removed, the steps to be removed are separated by `.` following the initial `^`.

```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
}
```



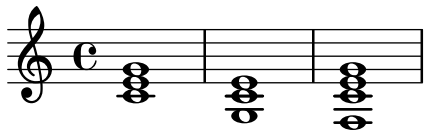
The modifier `sus` can be added to the modifier string to create suspended chords. This removes the 3rd step from the chord. Append either `2` or `4` to add the 2nd or 4th step to the chord. `sus` is equivalent to `^3`; `sus4` is equivalent to `.4^3`.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4^3
}
```



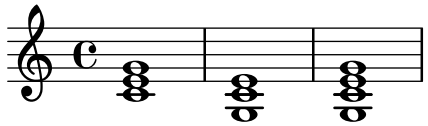
Inversions (putting a pitch other than the root on the bottom of the chord) and added bass notes can be specified by appending */pitch* to the chord.

```
\chordmode {
  c1 c/g c/f
}
```



A bass note that is part of the chord can be added, instead of moved as part of an inversion, by using */+pitch*.

```
\chordmode {
  c1 c/g c/+g
}
```



Chord modifiers that can be used to produce a variety of standard chords are shown in Sezione A.2 [Common chord modifiers], pagina 616.

Vedi anche
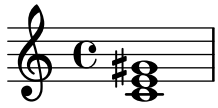
Notation Reference: Sezione A.2 [Common chord modifiers], pagina 616.

Snippets: Sezione “Chords” in *Frammenti di codice*.

Problemi noti e avvertimenti

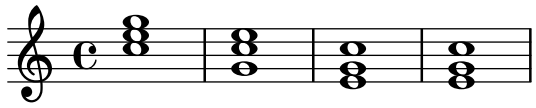
Each step can only be present in a chord once. The following simply produces the augmented chord, since `5+` is interpreted last.

```
\chordmode { c1:5.5-.5+ }
```



Only the second inversion can be created by adding a bass note. The first inversion requires changing the root of the chord.

```
\chordmode {
  c'1: c':/g e:6-3-^5 e:m6-^5
}
```



2.7.2 Displaying chords

Chords can be displayed by name, in addition to the standard display as notes on a staff.

Printing chord names

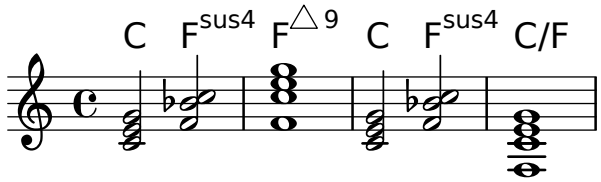
Chord names are printed in the `ChordNames` context:

```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```

C F G

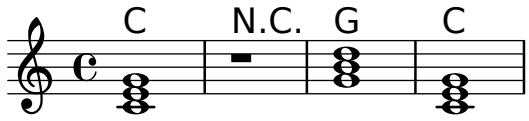
Chords can be entered as simultaneous notes or through the use of chord mode. The displayed chord name will be the same, regardless of the mode of entry, unless there are inversions or added bass notes:

```
chordmusic = \relative c' {
  <c e g>2 <f bes c>
  <f c' e g>1
  \chordmode {
    c2 f:sus4 c1:/f
  }
}
<<
  \new ChordNames {
    \chordmusic
  }
  {
    \chordmusic
  }
>>
```



Rests passed to a `ChordNames` context will cause the `noChordSymbol` markup to be displayed.

```
<<
  \new ChordNames \chordmode {
    c1
    r1
    g1
    c1
  }
  \chordmode {
    c1
    r1
    g1
    c1
  }
>>
```



`\chords { ... }` is a shortcut notation for `\new ChordNames { \chordmode { ... } }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```

C Fm G△

```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```

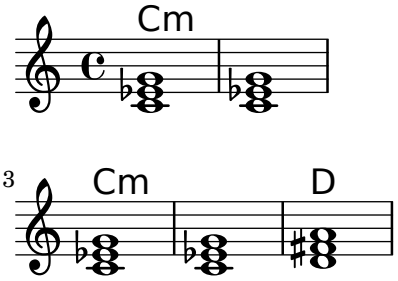
C Fm G△

Frammenti di codice selezionati

Showing chords at changes

Chord names can be displayed only at the start of lines and when the chord changes.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}
<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
  }
  \new Staff {
    \relative c' { \harmonies }
  }
>>
```

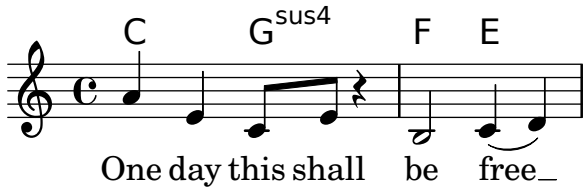


Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet:

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
```

```
  \addlyrics { One day this shall be free __ }
>>
```



Vedi anche

Music Glossary: Sezione “chord” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Writing music in parallel], pagina ⟨undefined⟩.

Snippets: Sezione “Chords” in *Frammenti di codice*.

Internals Reference: Sezione “ChordNames” in *Guida al Funzionamento Interno*, Sezione “ChordName” in *Guida al Funzionamento Interno*, Sezione “Chord_name_engraver” in *Guida al Funzionamento Interno*, Sezione “Volta_engraver” in *Guida al Funzionamento Interno*, Sezione “Bar_engraver” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

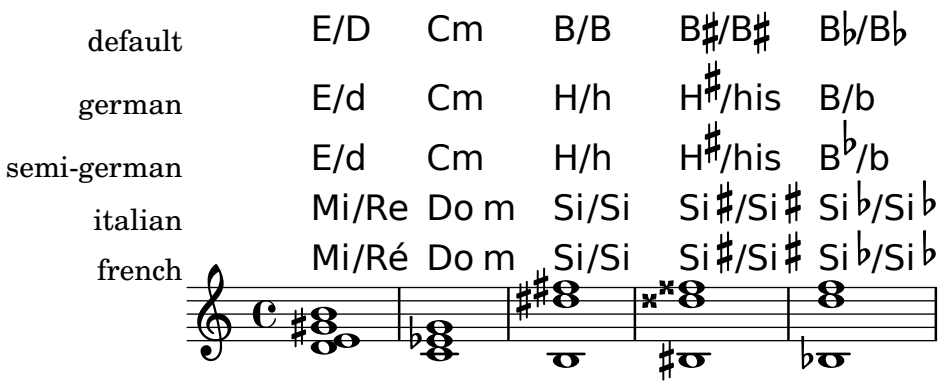
Chords containing inversions or altered bass notes are not named properly if entered using simultaneous music.

Customizing chord names

There is no unique system for naming chords. Different musical traditions use different names for the same set of chords. There are also different symbols displayed for a given chord name. The names and symbols displayed for chord names are customizable.

The basic chord name layout is a system for Jazz music, proposed by Klaus Ignatzek (see Sezione “Literature list” in *Saggio*). The chord naming system can be modified as described below. An alternate jazz chord system has been developed using these modifications. The Ignatzek and alternate Jazz notation are shown on the chart in Sezione A.1 [Chord name chart], pagina 615.

In addition to the different naming systems, different note names are used for the root in different languages. The predefined commands `\germanChords`, `\semiGermanChords`, `\italianChords` and `\frenchChords` set these variables. The effect is demonstrated here:



German songbooks may indicate minor chords as lowercase letters, without any *m* suffix. This can be obtained by setting the `chordNameLowercaseMinor` property:

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```



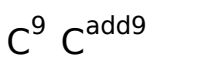
If none of the existing settings give the desired output, the chord name display can be tuned through the following properties.

`chordRootNamer`
: The chord name is usually printed as a letter for the root with an optional alteration. The transformation from pitch to letter is done by this function. Special note names (for example, the German 'H' for a B-chord) can be produced by storing a new function in this property.

`majorSevenSymbol`
: This property contains the markup object used to follow the output of `chordRootNamer` to identify a major 7 chord. Predefined options are `whiteTriangleMarkup` and `blackTriangleMarkup`.

`additionalPitchPrefix`
: When the chord name contains additional pitches, they can optionally be prefixed with some text. The default is no prefix, in order to avoid too much visual clutter, but for small numbers of additional pitches this can be visually effective.

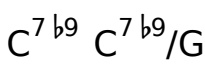
```
\new ChordNames {
  <c e g d'>    % add9
  \set additionalPitchPrefix = #"add"
  <c e g d'>    % add9
}
```

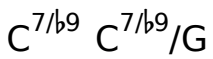


`chordNoteNamer`
: When the chord name contains additional pitches other than the root (e.g., an added bass note), this function is used to print the additional pitch. By default the pitch is printed using `chordRootNamer`. The `chordNoteNamer` property can be set to a specialized function to change this behavior. For example, the bass note can be printed in lower case.

`chordNameSeparator`
: Different parts of a chord name are normally separated by a small amount of horizontal space. By setting `chordNameSeparator`, you can use any desired markup for a separator. This does not affect the separator between a chord and its bass note; to customize that, use `slashChordSeparator`.

```
\chords {
  c4:7.9- c:7.9-/g
  \set chordNameSeparator = \markup { "/" }
  \break
  c4:7.9- c:7.9-/g
}
```

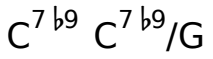


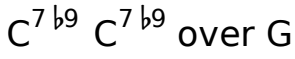


`slashChordSeparator`

Chords can be played over a bass note other than the conventional root of the chord. These are known as “inversions” or “slash chords”, because the default way of notating them is with a forward slash between the main chord and the bass note. Therefore the value of `slashChordSeparator` defaults to a forward slash, but you can change it to any markup you choose.

```
\chords {
  c4:7.9- c:7.9-/g
  \set slashChordSeparator = \markup { " over " }
  \break
  c4:7.9- c:7.9-/g
}
```





`chordNameExceptions`

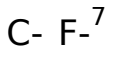
This property is a list of pairs. The first item in each pair is a set of pitches used to identify the steps present in the chord. The second item is a markup that will follow the `chordRootNamer` output to create the chord name.

`minorChordModifier`

Minor chords are often denoted via a ‘m’ suffix to the right of the root of the chord. However some idioms prefer other suffices, such as a minus sign.

```
\chords {
  c4:min f:min7
  \set minorChordModifier = \markup { "-" }
  \break
  c4:min f:min7
}
```





`chordPrefixSpacer`

The modifier for minor chords as determined by `minorChordModifier` is usually printed immediately to the right of the root of the chord. A spacer can be placed between the root and the modifier by setting `chordPrefixSpacer`. The spacer is not used when the root is altered.

Comandi predefiniti

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

Frammenti di codice selezionati

Chord name exceptions

The property `chordNameExceptions` can be used to store a list of special notations for specific chords.

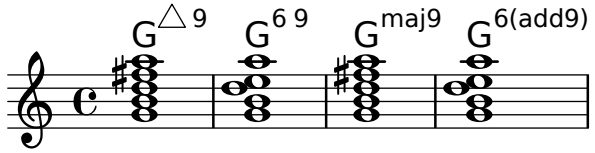
```
% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

% Convert music to list and prepend to existing exceptions.
chExceptions = #( append
  ( sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

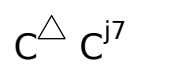
<< \context ChordNames \theMusic
   \context Voice \theMusic
>>
```



chord name major7

The layout of the major 7 can be tuned with `majorSevenSymbol`.

```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```

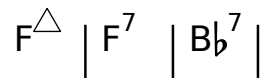


Adding bar lines to ChordNames context

To add bar line indications in the `ChordNames` context, add the `Bar_engraver`.

```
\new ChordNames \with {
  \override BarLine.bar-extent = #'(-2 . 2)
  \consists "Bar_engraver"
```

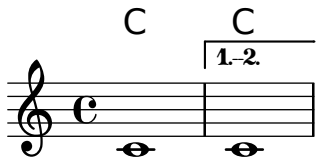
```
}
\chordmode {
  f1:maj7 f:7 bes:7
}
```



Volta below chords

By adding the `Volta_engraver` to the relevant staff, volte can be put under chords.

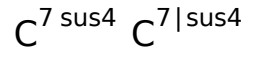
```
\score {
  <<
    \chords {
      c1
      c1
    }
    \new Staff \with {
      \consists "Volta_engraver"
    }
    {
      \repeat volta 2 { c'1 }
      \alternative { c' }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```



Changing chord separator

The separator between different parts of a chord name can be set to any markup.

```
\chords {
  c:7sus4
  \set chordNameSeparator
    = \markup { \typewriter | }
  c:7sus4
}
```



Vedi anche

Notation Reference: Sezione A.1 [Chord name chart], pagina 615, Sezione A.2 [Common chord modifiers], pagina 616.

Essay on automated music engraving: Sezione “Literature list” in *Saggio*.

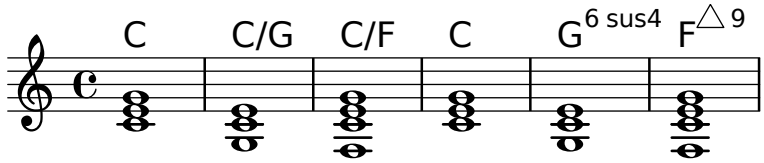
Installed Files: ‘`scm/chords-ignatzek.scm`’, ‘`scm/chord-entry.scm`’, ‘`ly/chord-modifier-init.ly`’.

Snippets: Sezione “Chords” in *Frammenti di codice*.

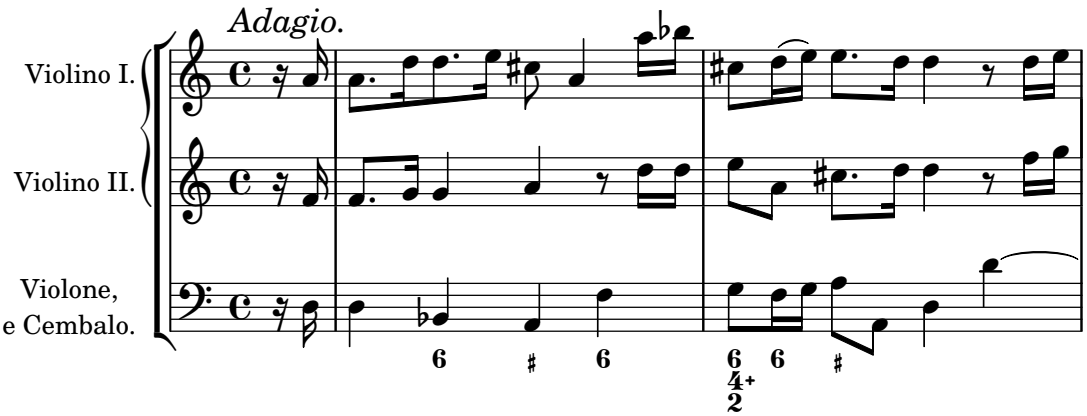
Problemi noti e avvertimenti
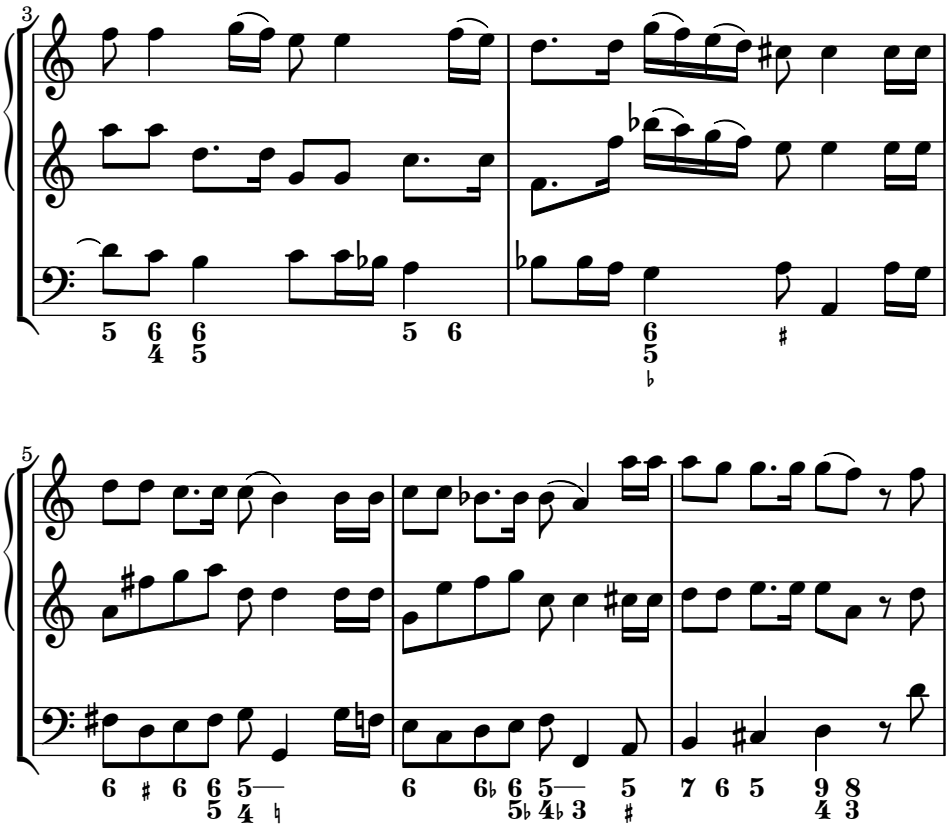
Chord names are determined from both the pitches that are present in the chord and the information on the chord structure that may have been entered in `\chordmode`. If the simultaneous pitches method of entering chords is used, undesired names result from inversions or bass notes.

```
myChords = \relative c' {
  \chordmode { c1 c/g c/f }
  <c e g>1 <g c e> <f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
>>
```



2.7.3 Figured bass



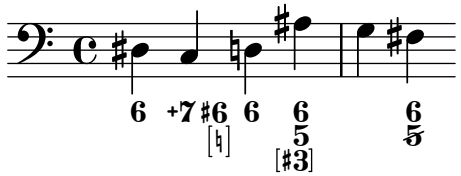


Figured bass notation can be displayed.

Introduction to figured bass

LilyPond has support for figured bass, also called thorough bass or basso continuo:

```
<<
  \new Voice { \clef bass dis4 c d ais g fis}
  \new FiguredBass {
    \figuremode {
      < 6 >4 < 7\+ >8 < 6+ [_!] >
      < 6 >4 <6 5 [3+] >
      < _ >4 < 6 5/>4
    }
  }
>>
```



The support for figured bass consists of two parts: there is an input mode, introduced by `\figuremode`, that accepts entry of bass figures, and there is a context named `FiguredBass` that takes care of displaying `BassFigure` objects. Figured bass can also be displayed in `Staff` contexts.

`\figures{ ... }` is a shortcut notation for `\new FiguredBass { \figuremode { ... } }`.

Although the support for figured bass may superficially resemble chord support, it is much simpler. `\figuremode` mode simply stores the figures and the `FiguredBass` context prints them as entered. There is no conversion to pitches.

Vedi anche

Music Glossary: Sezione “figured bass” in *Glossario Musicale*.

Snippets: Sezione “Chords” in *Frammenti di codice*.

Entering figured bass

`\figuremode` is used to switch the input mode to figure mode. More information on different input modes can be found at Sezione 5.4.1 [Input modes], pagina 583.

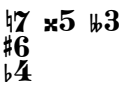
In figure mode, a group of bass figures is delimited by `<` and `>`. The duration is entered after the `>`.

```
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
```



Accidentals (including naturals) can be added to figures:

```
\figures {
  <7! 6+ 4-> <5++> <3-->
}
```



Augmented and diminished steps can be indicated:

```
\figures {
  <6\+ 5/> <7/>
}
```



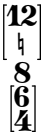
A backward slash through a figure (typically used for raised sixth steps) can be created:

```
\figures {
  <6> <6\\>
}
```



Vertical spaces and brackets can be included in figures:

```
\figures {
  <[12 _!] 8 [6  4]>
}
```



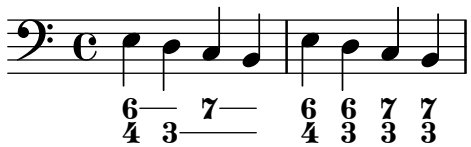
Any text markup can be inserted as a figure:

```
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
```
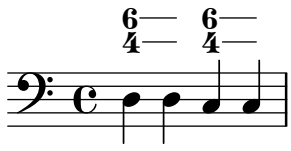


Continuation lines can be used to indicate repeated figures:

```
<<
  {
    \clef bass
    e4 d c b,
    e4 d c b,
  }
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 3> <7 3> <7 3>
    \bassFigureExtendersOff
    <6 4>4 <6 3> <7 3> <7 3>
  }
>>
```

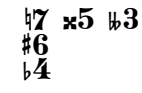


In this case, the extender lines replace existing figures, unless the continuation lines have been explicitly terminated.

```
<<
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 4> <6\! 4\!> <6 4>
  }
  {
    \clef bass
    d4 d c c
  }
>>
```



The table below summarizes the figure modifiers available.

Modifier	Purpose	Example
+, -, !	Accidentals	

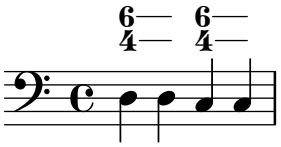
\+, / Augmented and diminished steps



\\ Raised sixth step



\! End of continuation line



Comandi predefiniti

`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

Frammenti di codice selezionati

Changing the positions of figured bass alterations

Accidentals and plus signs can appear before or after the numbers, depending on the `figuredBassAlterationDirection` and `figuredBassPlusDirection` properties.

```
\figures {
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5+> <6 4-> r
}
```



Vedi anche

Snippets: Sezione “Chords” in *Frammenti di codice*.

Internals Reference: Sezione “BassFigure” in *Guida al Funzionamento Interno*, Sezione “BassFigureAlignment” in *Guida al Funzionamento Interno*, Sezione “BassFigureLine” in *Guida al Funzionamento Interno*, Sezione “BassFigureBracket” in *Guida al Funzionamento Interno*, Sezione “BassFigureContinuation” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*.

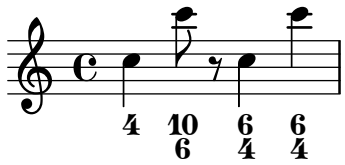
Displaying figured bass

Figured bass can be displayed using the `FiguredBass` context, or in most staff contexts.

When displayed in a `FiguredBass` context, the vertical location of the figures is independent of the notes on the staff.

```
<<
  \relative c'' {
    c4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
```

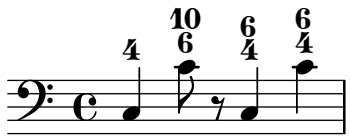
```
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
  }
>>
```



In the example above, the `FiguredBass` context must be explicitly instantiated to avoid creating a second (empty) staff.

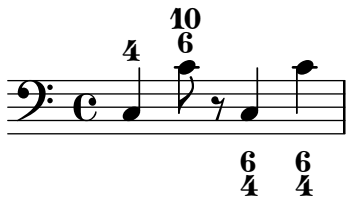
Figured bass can also be added to `Staff` contexts directly. In this case, the vertical position of the figures is adjusted automatically.

```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff"
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



When added in a `Staff` context, figured bass can be displayed above or below the staff.

```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    \bassFigureStaffAlignmentDown
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff"
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



Comandi predefiniti

`\bassFigureStaffAlignmentDown`, `\bassFigureStaffAlignmentUp`, `\bassFigureStaffAlignmentNeutral`.

Vedi anche
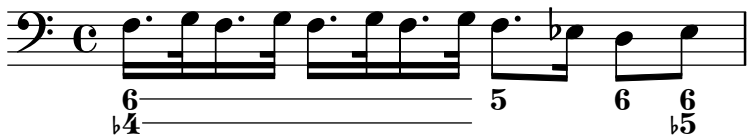
Snippets: Sezione “Chords” in *Frammenti di codice*.
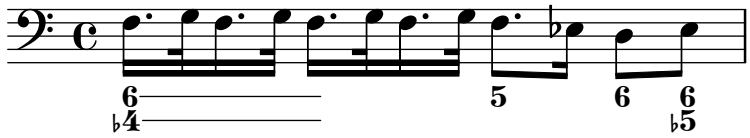
Internals Reference: Sezione “BassFigure” in *Guida al Funzionamento Interno*, Sezione “BassFigureAlignment” in *Guida al Funzionamento Interno*, Sezione “BassFigureLine” in *Guida al Funzionamento Interno*, Sezione “BassFigureBracket” in *Guida al Funzionamento Interno*, Sezione “BassFigureContinuation” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

To ensure that continuation lines work properly, it is safest to use the same rhythm in the figure line as in the bass line.

```
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are correct here, with the same rhythm as the bass
    \repeat unfold 4 { <6 4->16. <6 4->32 }
    <5>8. r16 <6>8 <6\! 5->
  }
>>
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are incorrect here, even though the timing is the same
    <6 4->4 <6 4->4
    <5>8. r16 <6>8 <6\! 5->
  }
>>
```





2.8 Contemporary music

From the beginning of the 20th Century there has been a massive expansion of compositional style and technique. New harmonic and rhythmic developments, an expansion of the pitch spectrum and the development of a wide range of new instrumental techniques have been accompanied by a parallel evolution and expansion of musical notation. The purpose of this section is to provide references and information relevant to working with these new notational techniques.

2.8.1 Pitch and harmony in contemporary music

This section highlights issues that are relevant to notating pitch and harmony in contemporary music.

References for pitch and harmony in contemporary music

- Standard quarter-tone notation is addressed in ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩.
- Non-standard key signatures are addressed in ⟨undefined⟩ [Key signature], pagina ⟨undefined⟩.
- Contemporary practises in displaying accidentals are addressed in ⟨undefined⟩ [Automatic accidentals], pagina ⟨undefined⟩.

Microtonal notation

Contemporary key signatures and harmony

2.8.2 Contemporary approaches to rhythm

This section highlights issues that are relevant to the notation of rhythm in contemporary music.

References for contemporary approaches to rhythm

- Compound time signatures are addressed in ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.
- Basic polymetric notation is addressed in ⟨undefined⟩ [Polymetric notation], pagina ⟨undefined⟩.
- Feathered beams are addressed in ⟨undefined⟩ [Feathered beams], pagina ⟨undefined⟩.
- Mensurstriche bar lines (bar lines between staves only) are addressed in ⟨undefined⟩ [Grouping staves], pagina ⟨undefined⟩.

Tuplets in contemporary music

Contemporary time signatures

Extended polymetric notation

Beams in contemporary music

Bar lines in contemporary music

2.8.3 Graphical notation

2.8.4 Contemporary scoring techniques

2.8.5 New instrumental techniques

2.8.6 Further reading and scores of interest

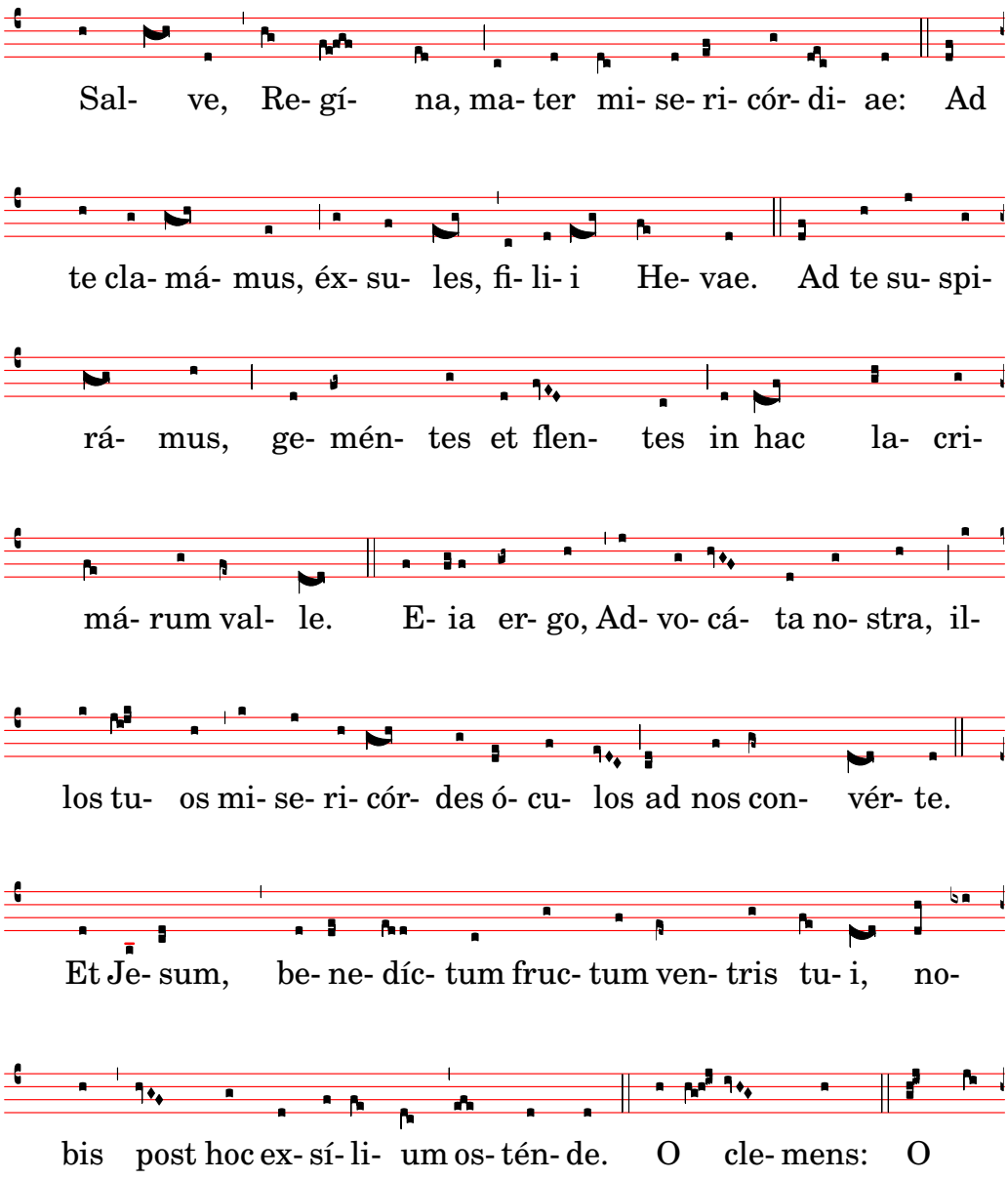
This section suggests books, musical examples and other resources useful in studying contemporary musical notation.

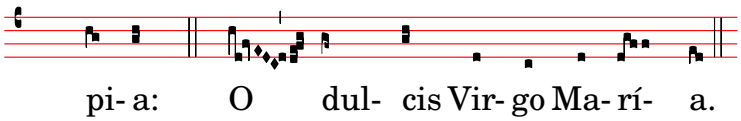
Books and articles on contemporary musical notation

- *Music Notation in the Twentieth Century: A Practical Guidebook* by Kurt Stone [W. W. Norton, 1980]
- *Music Notation: A Manual of Modern Practice* by Gardner Read [Taplinger, 1979]
- *Instrumentation and Orchestration* by Alfred Blatter [Schirmer, 2nd ed. 1997]

Scores and musical examples

2.9 Ancient notation





Support for ancient notation includes features for mensural notation, Gregorian chant notation, and Kievan square notation. These features can be accessed either by modifying style properties of graphical objects such as note heads and rests, or by using one of the pre-defined contexts for these styles.

Many graphical objects, such as note heads and flags, accidentals, time signatures, and rests, provide a `style` property, which can be changed to emulate several different styles of ancient notation. See

- [Mensural note heads], pagina 420,
- [Mensural accidentals and key signatures], pagina 422,
- [Mensural rests], pagina 421,
- [Mensural clefs], pagina 417,
- [Gregorian clefs], pagina 425,
- [Mensural flags], pagina 420,
- [Mensural time signatures], pagina 419.

Some notational concepts are introduced specifically for ancient notation,

- [Custodes], pagina 416,
- [Divisiones], pagina 426,
- [Ligatures], pagina 415.

Vedi anche

Music Glossary: Sezione “custos” in *Glossario Musicale*, Sezione “ligature” in *Glossario Musicale*, Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: [Mensural note heads], pagina 420, [Mensural accidentals and key signatures], pagina 422, [Mensural rests], pagina 421, [Gregorian clefs], pagina 425, [Mensural flags], pagina 420, [Mensural time signatures], pagina 419, [Custodes], pagina 416, [Divisiones], pagina 426, [Ligatures], pagina 415.

2.9.1 Overview of the supported styles

Three styles are available for typesetting Gregorian chant:

- *Editio Vaticana* is a complete style for Gregorian chant, following the appearance of the Solesmes editions, the official chant books of the Vatican since 1904. LilyPond has support for all the notational signs used in this style, including ligatures, *custodes*, and special signs such as the quilisma and the oriscus.
- The *Editio Medicaea* style offers certain features used in the Medicaea (or Ratisbona) editions which were used prior to the Solesmes editions. The most significant differences from the *Vaticana* style are the clefs, which have downward-slanted strokes, and the note heads, which are square and regular.
- The *Hufnagel* (“horseshoe nail”) or *Gothic* style mimics the writing style in chant manuscripts from Germany and Central Europe during the middle ages. It is named after the basic note shape (the *virga*), which looks like a small nail.

Three styles emulate the appearance of late-medieval and renaissance manuscripts and prints of mensural music:

- The *Mensural* style most closely resembles the writing style used in late-medieval and early renaissance manuscripts, with its small and narrow, diamond-shaped note heads and its rests which approach a hand-drawn style.

- The *Neomensural* style is a modernized and stylized version of the former: the note heads are broader and the rests are made up of straight lines. This style is particularly suited, e.g., for incipits of transcribed pieces of mensural music.
- The *Petrucci* style is named after Ottaviano Petrucci (1466-1539), the first printer to use movable type for music (in his *Harmonice musices odhecaton*, 1501). The style uses larger note heads than the other mensural styles.

Baroque and *Classical* are not complete styles but differ from the default style only in some details: certain note heads (Baroque) and the quarter rest (Classical).

Only the mensural style has alternatives for all aspects of the notation. Thus, there are no rests or flags in the Gregorian styles, since these signs are not used in plainchant notation, and the Petrucci style has no flags or accidentals of its own.

Each element of the notation can be changed independently of the others, so that one can use mensural flags, petrucci note heads, classical rests and vaticana clefs in the same piece, if one wishes.

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “flag” in *Glossario Musicale*.

2.9.2 Ancient notation—common features

Pre-defined contexts

For Gregorian chant and mensural notation, there are pre-defined voice and staff contexts available, which set all the various notation signs to values suitable for these styles. If one is satisfied with these defaults, one can proceed directly with note entry without worrying about the details on how to customize a context. See one of the pre-defined contexts `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice`, and `MensuralStaff`. See further

- [Gregorian chant contexts], pagina 424,
- [Mensural contexts], pagina 417.

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.
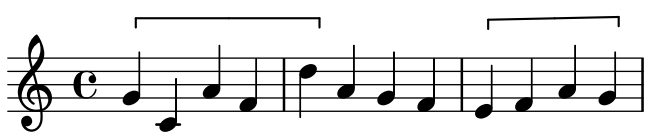
Notation Reference: [Gregorian chant contexts], pagina 424, [Mensural contexts], pagina 417.

Ligatures

A ligature is a graphical symbol that represents at least two distinct notes. Ligatures originally appeared in the manuscripts of Gregorian chant notation to denote ascending or descending sequences of notes on the same syllable. They are also used in mensural notation.

Ligatures are entered by *enclosing* them in `\[` and `\]`. Some ligature styles may need additional input syntax specific for this particular type of ligature. By default, the `LigatureBracket` engraver just puts a square bracket above the ligature.

```
\relative c'' {
  \[ g c, a' f d' \]
  a g f
  \[ e f a g \]
}
```



Two other ligature styles are available: the Vaticana for Gregorian chant, and the Mensural for mensural music (only white mensural ligatures are supported for mensural music, and with certain limitations). To use any of these styles, the default `Ligature_bracket_engraver` has to be replaced with one of the specialized ligature engravers in the `Voice` context, as explained in [White mensural ligatures], pagina 423 and [Gregorian square neume ligatures], pagina 428.

Vedi anche

Music Glossary: Sezione "ligature" in *Glossario Musicale*.

Notation Reference: [White mensural ligatures], pagina 423, [Gregorian square neume ligatures], pagina 428.

Problemi noti e avvertimenti

Ligatures need special spacing that has not yet been implemented. As a result, there is too much space between ligatures most of the time, and line breaking often is unsatisfactory. Also, lyrics do not correctly align with ligatures.

Accidentals must not be printed within a ligature, but instead need to be collected and printed in front of it.

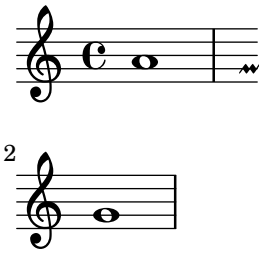
The syntax still uses the deprecated infix style `\[ music expr \]`. For consistency reasons, it will eventually be changed to postfix style `note\[ ... note\]`.

Custodes

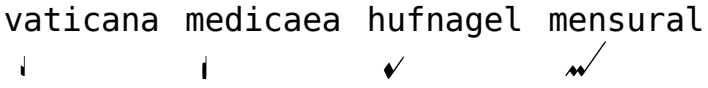
A *custos* (plural: *custodes*; Latin word for "guard") is a symbol that appears at the end of a staff. It anticipates the pitch of the first note of the following line, thus helping the performer to manage line breaks during performance.

Custodes were frequently used in music notation until the seventeenth century. Nowadays, they have survived only in a few particular forms of musical notation such as contemporary editions of Gregorian chant like the *Editio Vaticana*. There are different custos glyphs used in different flavors of notational style.

For typesetting custodes, just put a `Custos_engraver` into the `Staff` context when declaring the `\layout` block, and change the style of the custos with an `\override` if desired, as shown in the following example:



The custos glyph is selected by the `style` property. The styles supported are `vaticana`, `medicaea`, `hufnagel`, and `mensural`. They are demonstrated in the following fragment.



Vedi anche

Music Glossary: Sezione "custos" in *Glossario Musicale*.

Snippets: Sezione "Ancient notation" in *Frammenti di codice*.

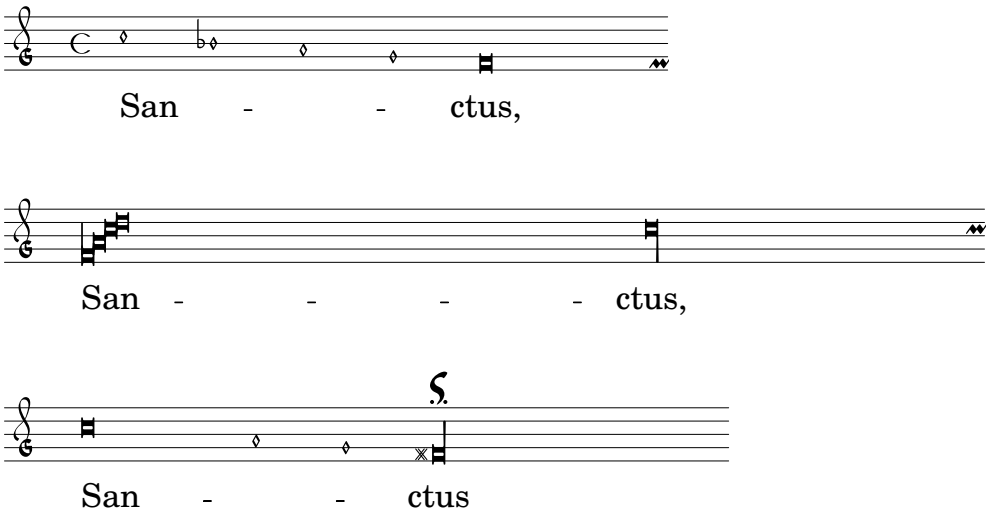
Internals Reference: Sezione "Custos" in *Guida al Funzionamento Interno*.

2.9.3 Typesetting mensural music

Mensural contexts

The predefined `MensuralVoice` and `MensuralStaff` contexts can be used to engrave a piece in mensural style. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

```
\score {
  <<
    \new MensuralVoice = "discantus" \relative c'' {
      \hide Score.BarNumber {
        c1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c\breve d\melismaEnd \]
        c\longa
        c\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```



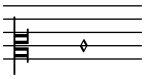
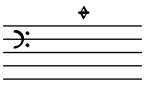
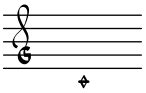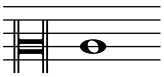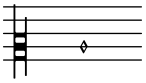
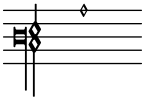
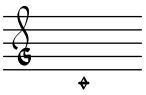
Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Mensural clefs

The following table shows all mensural clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. You can manually force a clef glyph to be typeset on an arbitrary line, as described in ⟨undefined⟩ [Clef], pagina ⟨undefined⟩. The note printed to the right side of each clef in the example column denotes the `c'` with respect to that clef.

Petrucci used C clefs with differently balanced left-side vertical beams, depending on which staff line it is printed.

Description	Supported Clefs	Example
mensural C clef	`mensural-c1`, `mensural-c2`, `mensural-c3`, `mensural-c4`, `mensural-c5`	
mensural F clef	`mensural-f`	
mensural G clef	`mensural-g`	
black mensural C clef	`blackmensural-c1`, `blackmensural-c2`, `blackmensural-c3`, `blackmensural-c4`, `blackmensural-c5`	
neomensural C clef	`neomensural-c1`, `neomensural-c2`, `neomensural-c3`, `neomensural-c4`	
petrucci style C clefs, for use on different staff lines (the example shows the 2nd staff line C clef)	`petrucci-c1`, `petrucci-c2`, `petrucci-c3`, `petrucci-c4`, `petrucci-c5`	
petrucci style F clefs, for use on different staff lines (the example shows the 3rd staff line F clef)	`petrucci-f3`, `petrucci-f4`, `petrucci-f5`	
petrucci style G clef	`petrucci-g`	

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “clef” in *Glossario Musicale*.

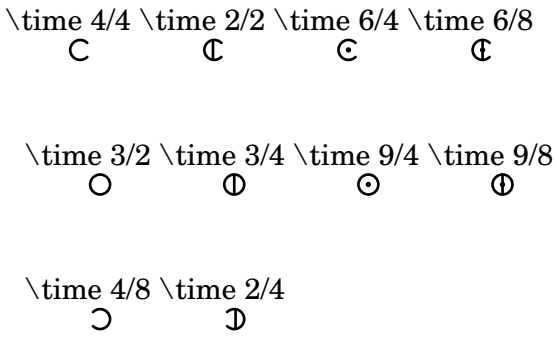
Notation Reference: ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

Problemi noti e avvertimenti

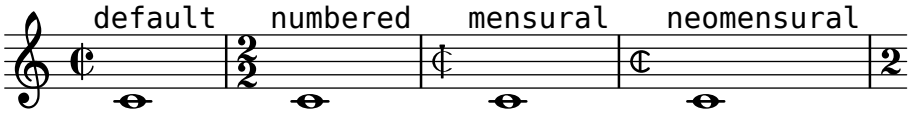
The mensural g clef is mapped to the Petrucci g clef.

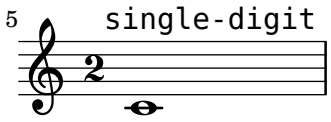
Mensural time signatures

There is limited support for mensuration signs (which are similar to, but not exactly the same as time signatures). The glyphs are hard-wired to particular time fractions. In other words, to get a particular mensuration sign with the `\time n/m` command, `n` and `m` have to be chosen according to the following table



Use the `style` property of grob `TimeSignature` to select ancient time signatures. Supported styles are `neomensural` and `mensural`. The above table uses the `neomensural` style. The following examples show the differences in style:





⟨undefined⟩ [Time signature], pagina ⟨undefined⟩, gives a general introduction to the use of time signatures.

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.

Problemi noti e avvertimenti

Ratios of note durations cannot change with the time signature, as those are not constant. For example, the ratio of 1 breve = 3 semibreves (*tempus perfectum*) can be made by hand, by setting

```
breveTP = #(ly:make-duration -1 0 3/2)
...
{ c\breveTP f1 }
```

This sets `breveTP` to 3/2 times 2 = 3 times a whole note.

The `mensural68alt` and `neomensural68alt` symbols (alternate symbols for 6/8) are not addressable with `\time`. Use `\markup {\musicglyph #"timesig.mensural68alt" }` instead.

Mensural note heads

For ancient notation, a note head style other than the `default` style may be chosen. This is accomplished by setting the `style` property of the `NoteHead` object to `baroque`, `neomensural`, `mensural`, `petrucci`, `blackpetrucci` or `semipetrucci`.

The `baroque` style differs from the `default` style by:

- Providing a `maxima` note head, and
- Using a square shape for `\breve` note heads.

The `neomensural`, `mensural`, and `petrucci` styles differ from the `baroque` style by:
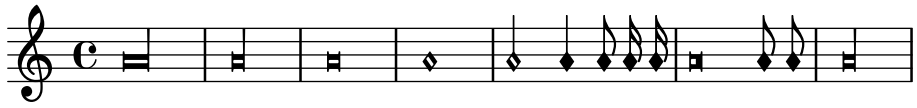
- Using rhomboidal heads for semibreves and all smaller durations, and
- Centering the stems on the note heads.

The `blackpetrucci` style produces note heads usable in black mensural notation or coloratio sections in white mensural notation. Because note head style does not influence flag count, in this style a semiminima should be notated as `a8*2`, not `a4`, otherwise it will look like a minima. The multiplyer can be different if coloratio is used e.g. to notate triplets.

Use `semipetrucci` style to draw half-colored note heads (breves, longas and maximas).

The following example demonstrates the `petrucci` style:

```
\set Score.skipBars = ##t
\autoBeamOff
\override NoteHead.style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead.style = #'petrucci
a'\longa
```



Sezione A.9 [Note head styles], pagina 653, gives an overview of all available note head styles.

Vedi anche

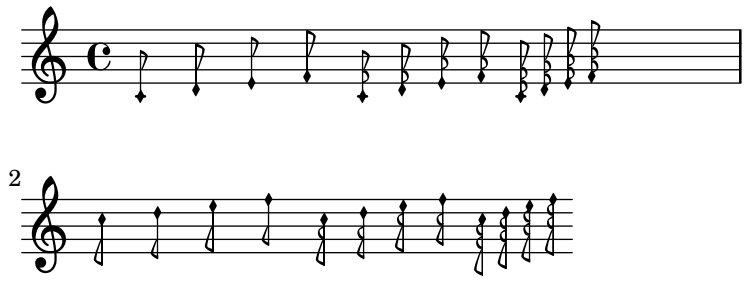
Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “note head” in *Glossario Musicale*.

Notation Reference: Sezione A.9 [Note head styles], pagina 653.

Mensural flags

Use the `flag-style` property of grob `Stem` to select ancient flags. Besides the `default` flag style, only the `mensural` style is supported.

```
\override Flag.style = #'mensural
\override Stem.thickness = #1.0
\override NoteHead.style = #'mensural
\autoBeamOff
c8 d e f c16 d e f c32 d e f s8
c'8 d e f c16 d e f c32 d e f
```



Note that the innermost flare of each mensural flag is vertically aligned with a staff line.

There is no particular flag style for neo-mensural or Petrucci notation. There are no flags in Gregorian chant notation.

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “flag” in *Glossario Musicale*.

Problemi noti e avvertimenti

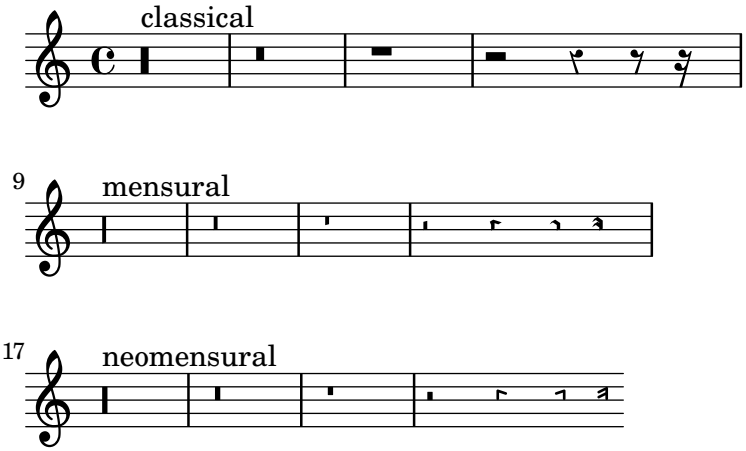
Vertically aligning each flag with a staff line assumes that stems always end either exactly on or exactly in the middle of two staff lines. This may not always be true when using advanced layout features of classical notation (which however are typically out of scope for mensural notation).

Mensural rests

Use the `style` property of grob `Rest` to select ancient rests. Supported styles are `classical`, `neomensural`, and `mensural`. `classical` differs from the `default` style only in that the quarter rest looks like a horizontally mirrored 8th rest. The `mensural` and the `neomensural` styles mimic the appearance of rests in manuscripts and prints up to the 16th century.

The following example demonstrates the `mensural` and `neomensural` styles:

```
\set Score.skipBars = ##t
\override Rest.style = #'classical
r\longa^"classical" r\breve r1 r2 r4 r8 r16 s \break
\override Rest.style = #'mensural
r\longa^"mensural" r\breve r1 r2 r4 r8 r16 s \break
\override Rest.style = #'neomensural
r\longa^"neomensural" r\breve r1 r2 r4 r8 r16
```



There are no 32th and 64th rests specifically for the mensural or neo-mensural style. Instead, the rests from the default style will be taken.

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Rests], pagina ⟨undefined⟩.

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

Problemi noti e avvertimenti

The glyph for the maxima rest in mensural style is actually a perfect longa rest; use two (or three) longa rests to print a maxima rest. Longa rests are not grouped automatically, so have to be done manually by using pitched rests.

Mensural accidentals and key signatures

The `mensural` style provides a sharp and a flat sign different from the default style. If called for, the natural sign will be taken from the `vaticana` style.

mensural

The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “Pitch names” in *Glossario Musicale*, Sezione “accidental” in *Glossario Musicale*, Sezione “key signature” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Pitches], pagina ⟨undefined⟩, ⟨undefined⟩ [Accidentals], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic accidentals], pagina ⟨undefined⟩, ⟨undefined⟩ [Key signature], pagina ⟨undefined⟩.
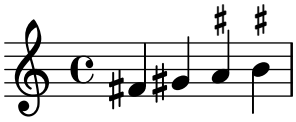
Internals Reference: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

Annotational accidentals (*musica ficta*)

In European music from before about 1600, singers were expected to chromatically alter notes at their own initiative according to certain rules. This is called *musica ficta*. In modern transcriptions, these accidentals are usually printed over the note.
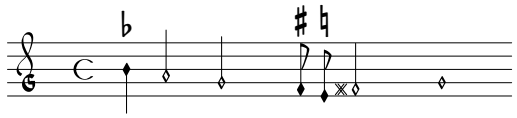
Support for such suggested accidentals is included, and can be switched on by setting `suggestAccidentals` to true.

```
fis gis
\set suggestAccidentals = ##t
ais bis
```



This will treat *every* subsequent accidental as *musica ficta* until it is unset with `\set suggestAccidentals = ##f`. A more practical way is to use `\once \set suggestAccidentals = ##t`, which can even be defined as a convenient shorthand:

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative c''
  \new MensuralVoice  {
    \once \set suggestAccidentals = ##t
    bes4 a2 g2 \ficta fis8 \ficta e! fis2 g1
  }
}
```



Vedi anche

Internals Reference: Sezione “Accidental_engraver” in *Guida al Funzionamento Interno*, Sezione “AccidentalSuggestion” in *Guida al Funzionamento Interno*.

White mensural ligatures

There is limited support for white mensural ligatures.

To engrave white mensural ligatures, in the layout block, replace the `Ligature_bracket_engraver` with the `Mensural_ligature_engraver` in the `Voice` context:

```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Mensural_ligature_engraver"
  }
}
```

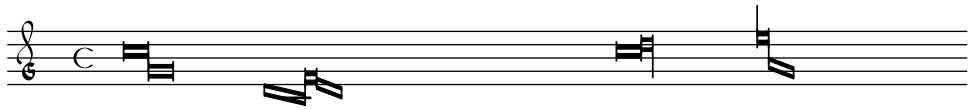
There is no additional input language to describe the shape of a white mensural ligature. The shape is rather determined solely from the pitch and duration of the enclosed notes. While this approach may take a new user a while to get accustomed to, it has the great advantage that the full musical information of the ligature is known internally. This is not only required for correct MIDI output, but also allows for automatic transcription of the ligatures.

At certain places two consecutive notes can be represented either as two squares or as an oblique parallelogram (flexa shape). In such cases the default is the two squares, but a flexa can be required by setting the `ligature-flexa` property of the *second* note head. The length of a flexa can be set by the note head property `flexa-width`.
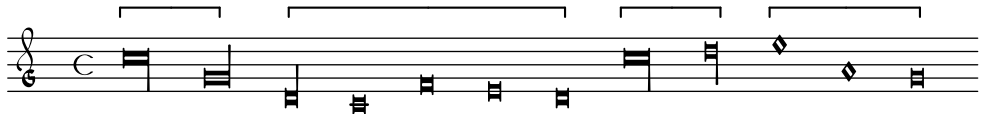
For example,

```
\score {
  \relative c' {
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-g"
    \[ c'\maxima g \]
    \[ d\longa
       \override NoteHead.ligature-flexa = ##t
       \once \override NoteHead.flexa-width = #3.2
```

```
      c\breve f e d \]
    \[ c'\maxima d\longa \]
    \[ e1 a, g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove "Ligature_bracket_engraver"
      \consists "Mensural_ligature_engraver"
    }
  }
}
```



Without replacing `Ligature_bracket_engraver` with `Mensural_ligature_engraver`, the same music transcribes to the following



Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [Gregorian square neume ligatures], pagina 428, [Ligatures], pagina 415.

Problemi noti e avvertimenti

Horizontal spacing of ligatures is poor. Accidentals may collide with previous notes.

2.9.4 Typesetting Gregorian chant

When typesetting a piece in Gregorian chant notation, the `Vaticana_ligature_engraver` automatically selects the proper note heads, so there is no need to explicitly set the note head style. Still, the note head style can be set, e.g., to `vaticana_punctum` to produce punctum neumes. Similarly, the `Mensural_ligature_engraver` automatically assembles mensural ligatures.

Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

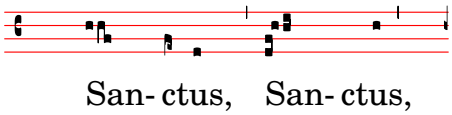
Notation Reference: [White mensural ligatures], pagina 423, [Ligatures], pagina 415.
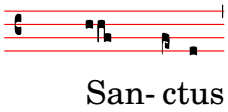
Gregorian chant contexts

The predefined `VaticanaVoiceContext` and `VaticanaStaffContext` can be used to engrave a piece of Gregorian chant in the style of the Editio Vaticana. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

```
\include "gregorian.ly"
\score {
  <<
    \new VaticanaVoice = "cantus" {
```

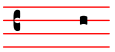
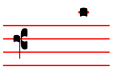
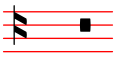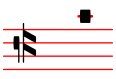
```
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new Lyrics \lyricsto "cantus" {
      San- ctus, San- ctus, San- ctus
    }
  >>
}
```





Gregorian clefs

The following table shows all Gregorian clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. Still, you can manually force a clef glyph to be typeset on an arbitrary line, as described in ⟨undefined⟩ [Clef], pagina ⟨undefined⟩. The note printed to the right side of each clef in the example column denotes the `c'` with respect to that clef.

Description	**Supported Clefs**	**Example**
Editio Vaticana style do clef	`vaticana-do1`, `vaticana-do2`, `vaticana-do3`	
Editio Vaticana style fa clef	`vaticana-fa1`, `vaticana-fa2`	
Editio Medicaea style do clef	`medicaea-do1`, `medicaea-do2`, `medicaea-do3`	
Editio Medicaea style fa clef	`medicaea-fa1`, `medicaea-fa2`	

hufnagel style do clef	`hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3`	
hufnagel style fa clef	`hufnagel-fa1`, `hufnagel-fa2`	
hufnagel style combined do/fa clef	`hufnagel-do-fa`	

Vedi anche

Music Glossary: Sezione “clef” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

Gregorian accidentals and key signatures

Accidentals for the three different Gregorian styles are available:

vaticana medicaea hufnagel



As shown, not all accidentals are supported by each style. When trying to access an unsupported accidental, LilyPond will switch to a different style.

The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

Vedi anche

Music Glossary: Sezione “accidental” in *Glossario Musicale*, Sezione “key signature” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Pitches], pagina ⟨undefined⟩, ⟨undefined⟩ [Accidentals], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic accidentals], pagina ⟨undefined⟩, ⟨undefined⟩ [Key signature], pagina ⟨undefined⟩.

Internals Reference: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

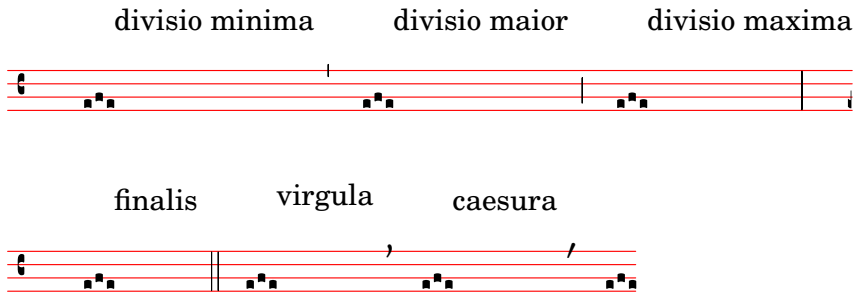
Divisiones

There are no rests in Gregorian chant notation; instead, it uses [Divisiones], pagina 426.

A *divisio* (plural: *divisiones*; Latin word for ‘division’) is a staff context symbol that is used to indicate the phrase and section structure of Gregorian music. The musical meaning of *divisio minima*, *divisio maior*, and *divisio maxima* can be characterized as short, medium, and long pause, somewhat like the breath marks from ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩. The *finalis* sign not only marks the end of a chant, but is also frequently used within a single antiphonal/responsorial chant to mark the end of each section.

To use divisiones, include the file ‘`gregorian.ly`’. It contains definitions that you can apply by just inserting `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, and `\finalis` at proper

places in the input. Some editions use *virgula* or *caesura* instead of divisio minima. Therefore, ‘`gregorian.ly`’ also defines `\virgula` and `\caesura`



Comandi predefiniti

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

Vedi anche

Music Glossary: Sezione “caesura” in *Glossario Musicale*, Sezione “divisio” in *Glossario Musicale*.

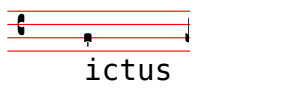
Notation Reference: ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩.

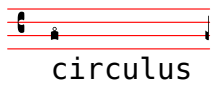
Installed Files: ‘`ly/gregorian.ly`’.

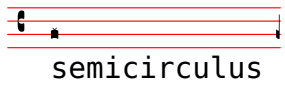
Gregorian articulation signs

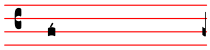
In addition to the standard articulation signs described in section ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩, articulation signs specifically designed for use with notation in *Editio Vaticana* style are provided.

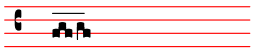
```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \override TextScript.font-family = #'typewriter
    \override TextScript.font-shape = #'upright
    \override Script.padding = #-0.1
    a\ictus_"ictus " \bar "" \break
    a\circulus_"circulus " \bar "" \break
    a\semicirculus_"semicirculus " \bar "" \break
    a\accentus_"accentus " \bar "" \break
    \[ a_"episema" \episemInitium \pes b \flexa a b \episemFinis \flexa a \]
  }
}
```











Vedi anche

Notation Reference: ⟨undefined⟩ [Articulations and ornamentations], pagina ⟨undefined⟩.

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

Internals Reference: Sezione “Episema” in *Guida al Funzionamento Interno*, Sezione “EpisemaEvent” in *Guida al Funzionamento Interno*, Sezione “Episema_engraver” in *Guida al Funzionamento Interno*, Sezione “Script” in *Guida al Funzionamento Interno*, Sezione “ScriptEvent” in *Guida al Funzionamento Interno*, Sezione “Script_engraver” in *Guida al Funzionamento Interno*.
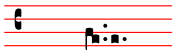
Problemi noti e avvertimenti

Some articulations are vertically placed too closely to the corresponding note heads.

Augmentum dots (*morae*)

Augmentum dots, also called *morae*, are added with the music function `\augmentum`. Note that `\augmentum` is implemented as a unary music function rather than as head prefix. It applies to the immediately following music expression only. That is, `\augmentum \virga c` will have no visible effect. Instead, say `\virga \augmentum c` or `\augmentum {\virga c}`. Also note that you can say `\augmentum {a g}` as a shortcut for `\augmentum a \augmentum g`.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩.

Internals Reference: Sezione “BreathingSign” in *Guida al Funzionamento Interno*.

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

Gregorian square neume ligatures

There is limited support for Gregorian square neumes notation (following the style of the Editio Vaticana). Core ligatures can already be typeset, but essential issues for serious typesetting are still lacking, such as (among others) horizontal alignment of multiple ligatures, lyrics alignment, and proper handling of accidentals.

The support for Gregorian neumes is enabled by `\include`ing ‘`gregorian.ly`’ at the beginning of the file. This makes available a number of extra commands to produce the neume symbols used in plainchant notation.

Note heads can be *modified* and/or *joined*.

- The shape of the note head can be modified by *prefixing* the note name with any of the following commands: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Ligatures, properly speaking (i.e. notes joined together), are produced by placing one of the joining commands `\pes` or `\flexa`, for upwards and downwards movement, respectively, *between* the notes to be joined.

A note name without any qualifiers will produce a *punctum*. All other neumes, including the single-note neumes with a different shape such as the *virga*, are in principle considered as ligatures and should therefore be placed between `\[...\]`.

Single-note neumes:

- The *punctum* is the basic note shape (in the *Vaticana* style: a square with some curvation for typographical finesse). In addition to the regular *punctum*, there is also the oblique *punctum inclinatum*, produced with the prefix `\inclinatum`. The regular *punctum* can be modified with `\cavum`, which produces a hollow note, and `\linea`, which draws vertical lines on either side of the note.
- The *virga* has a descending stem on the right side. It is produced by the modifier `\virga`.

Ligatures

Unlike most other neumes notation systems, the typographical appearance of ligatures is not directly dictated by the input commands, but follows certain conventions dependent on musical meaning. For example, a three-note ligature with the musical shape low-high-low, such as `\[ a \pes b \flexa g \]`, produces a Torculus consisting of three Punctum heads, while the shape high-low-high, such as `\[ a \flexa g \pes b \]`, produces a Porrectus with a curved flexa shape and only a single Punctum head. There is no command to explicitly typeset the curved flexa shape; the decision of when to typeset a curved flexa shape is based on the musical input. The idea of this approach is to separate the musical aspects of the input from the notation style of the output. This way, the same input can be reused to typeset the same music in a different style of Gregorian chant notation.

Liquescent neumes

Another main category of notes in Gregorian chant is the so-called liquescent neumes. They are used under certain circumstances at the end of a syllable which ends in a 'liquescent' letter, i.e. the sounding consonants that can hold a tone (the nasals, l, r, v, j, and their diphthong equivalents). Thus, the liquescent neumes are never used alone (although some of them can be produced), and they always fall at the end of a ligature.

Liquescent neumes are represented graphically in two different, more or less interchangeable ways: with a smaller note or by 'twisting' the main note upwards or downwards. The first is produced by making a regular `pes` or `flexa` and modifying the shape of the second note: `\[ a \pes \deminutum b \]` , the second by modifying the shape of a single-note neume with `\auctum` and one of the direction markers `\descendens` or `\ascendens`, e.g., `\[ \auctum \descendens a \]` .

Special signs

A third category of signs is made up of a small number of signs with a special meaning (which, incidentally, in most cases is only vaguely known): the *quilisma*, the *oriscus*, and the *strophicus*. These are all produced by prefixing a note name with the corresponding modifier, `\quilisma`, `\oriscus`, or `\stropha`.

Virtually, within the ligature delimiters `\[` and `\]`, any number of heads may be accumulated to form a single ligature, and head prefixes like `\pes`, `\flexa`, `\virga`, `\inclinatum`, etc. may be mixed in as desired. The use of the set of rules that underlies the construction of the ligatures in the above table is accordingly extrapolated. This way, infinitely many different ligatures can be created.

Note that the use of these signs in the music itself follows certain rules, which are not checked by LilyPond. E.g., the *quilisma* is always the middle note of an ascending ligature, and usually falls on a half-tone step, but it is perfectly possible, although incorrect, to make a single-note quilisma.

In addition to the note signs, '`gregorian.ly`' also defines the commands `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ`, and `\IIJ`, that will produce the corresponding characters, e.g., for use in lyrics, as section markers, etc. These commands use special Unicode characters and will only work if a font is used which supports them.

The following table shows a limited, but still representative pool of Gregorian ligatures, together with the code fragments that produce the ligatures. The table is based on the extended neumes table of the 2nd volume of the Antiphonale Romanum (*Liber Hymnarius*), published 1983 by the monks of Solesmes. The first column gives the name of the ligature, with the main form in boldface and the liquescent forms in italics. The third column shows the code fragment that produces this ligature, using `g`, `a`, and `b` as example pitches.

Single-note neums

Basic and *Liquescent* forms	**Output**	**LilyPond code**
Punctum		`\[ b \]`
		`\[ \cavum b \]`
		`\[ \linea b \]`
Punctum Auctum Ascendens		`\[ \auctum \ascendens b \]`
		
Punctum Auctum Descendens		`\[ \auctum \descendens b \]`
		
Punctum inclinatum		`\[ \inclinatum b \]`
		

Punctum Inclinatum Auctum

```
\[ \inclinatum \auctum b \]
```



Punctum Inclinatum Parvum

```
\[ \inclinatum \deminutum b \]
```


Virga



Two-note ligatures

Clivis vel Flexa

```
\[ b \flexa g \]
```



Clivis Aucta Descendens

```
\[ b \flexa \auctum \descendens
g \]
```



Clivis Aucta Ascendens

```
\[ b \flexa \auctum \ascendens
g \]
```



Cephalicus

```
\[ b \flexa \deminutum g \]
```



Podatus/Pes

```
\[ g \pes b \]
```



Pes Auctus Descendens

```
\[ g \pes \auctum \descendens b
\]
```



Pes Auctus Ascendens



```
\[ g \pes \auctum \ascendens b
\]
```

Epiphonus



```
\[ g \pes \deminutum b \]
```

Pes Initio Debilis



```
\[ \deminutum g \pes b \]
```

Pes Auctus Descendens Initio Debilis



```
\[ \deminutum g \pes \auctum
\descendens b \]
```

Multi-note ligatures

Torculus



```
\[ a \pes b \flexa g \]
```

Torculus Auctus Descendens



```
\[ a \pes b \flexa \auctum
\descendens g \]
```

Torculus Deminutus



```
\[ a \pes b \flexa \deminutum g
\]
```

Torculus Initio Debilis



```
\[ \deminutum a \pes b \flexa g
\]
```

Torculus Auctus Descendens Initio Debilis



```
\[ \deminutum a \pes b \flexa
\auctum \descendens g \]
```

Torculus Deminutus Initio Debilis


```
\[ \deminutum a \pes b \flexa
\deminutum g \]
```

Porrectus


```
\[ a \flexa g \pes b \]
```

Porrectus Auctus Descendens


```
\[ a \flexa g \pes \auctum
\descendens b \]
```

Porrectus Deminutus


```
\[ a \flexa g \pes \deminutum b
\]
```

Climacus


```
\[ \virga b \inclinatum a
\inclinatum g \]
```

Climacus Auctus


```
\[ \virga b \inclinatum a
\inclinatum \auctum g \]
```

Climacus Deminutus


```
\[ \virga b \inclinatum a
\inclinatum \deminutum g \]
```

Scandicus


```
\[ g \pes a \virga b \]
```

Scandicus Auctus Descendens


```
\[ g \pes a \pes \auctum
\descendens b \]
```

Scandicus Deminutus

```
\[ g \pes a \pes \deminutum b \]
```



Special Signs

Quilisma

```
\[ g \pes \quilisma a \pes b \]
```



Quilisma Pes Auctus Descendens

```
\[ \quilisma g \pes \auctum
\descendens b \]
```



Oriscus

```
\[ \oriscus b \]
```


Pes Quassus

```
\[ \oriscus g \pes \virga b \]
```



Pes Quassus Auctus Descendens

```
\[ \oriscus g \pes \auctum
\descendens b \]
```



Salicus

```
\[ g \oriscus a \pes \virga b \]
```



Salicus Auctus Descendens

```
\[ g \oriscus a \pes \auctum
\descendens b \]
```



(Apo)stropha

```
\[ \stropha b \]
```



Stropha Aucta

```
\[ \stropha \auctum b \]
```



Bistropha

```
\[ \stropha b \stropha b \]
```



Tristropha

```
\[ \stropha b \stropha b
\stropha b \]
```



Trigonus

```
\[ \stropha b \stropha b
\stropha a \]
```



Comandi predefiniti

The following head prefixes are supported: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Head prefixes can be accumulated, though restrictions apply. For example, either `\descendens` or `\ascendens` can be applied to a head, but not both to the same head.

Two adjacent heads can be tied together with the `\pes` and `\flexa` infix commands for a rising and falling line of melody, respectively.

Use the unary music function `\augmentum` to add augmentum dots.

Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [Gregorian square neume ligatures], pagina 428, [White mensural ligatures], pagina 423, [Ligatures], pagina 415.

Problemi noti e avvertimenti

When an `\augmentum` dot appears at the end of the last staff within a ligature, it is sometimes vertically placed wrong. As a workaround, add an additional skip note (e.g., `s8`) as last note of the staff.

`\augmentum` should be implemented as a head prefix rather than a unary music function, such that `\augmentum` can be intermixed with head prefixes in arbitrary order.
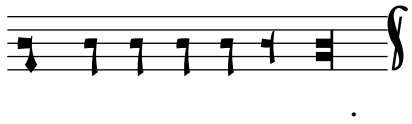
2.9.5 Typesetting Kievan square notation

Kievan contexts

As with Mensural and Gregorian notation, the predefined `KievanVoice` and `KievanStaff` contexts can be used to engrave a piece in square notation. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant:

```
\score {
```

```
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
        c4 c c c c2 b\longa
        \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Го -- спо -- ди по -- ми -- луй.
    }
  >>
}
```



Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*.
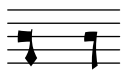
Problemi noti e avvertimenti

LilyPond supports Kievan notation of the Synodal style, as used in the corpus of chantbooks printed by the Russian Holy Synod in the 1910’s and recently reprinted by the Moscow Patriarchate Publishing House. LilyPond does not support the older (less common) forms of Kievan notation that were used in Galicia to notate Rusyn plainchant.

Kievan clefs

There is only one clef used in Kievan notation (the Tse-fa-ut Clef). It is used to indicate the position of `c`:

```
\clef "kievan-do"
\kievanOn
c
```



Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*, Sezione “clef” in *Glossario Musicale*.
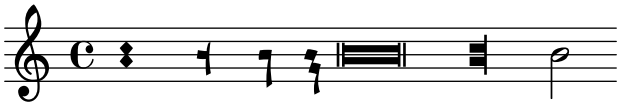
Notation Reference: ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

Kievan notes

For Kievan square notation, the appropriate note head style needs to be chosen and the flags and stems need to be turned off. This is accomplished by calling the `\kievanOn` function, which sets the appropriate properties of the note head, stems, and flags. Once Kievan note heads are not needed, these properties can be reverted by calling the `\kievanOff` function.

The Kievan final note, which usually comes at the end of a piece of music, may be selected by setting the duration to `\longa`. The Kievan recitative mark, used to indicate the chanting of several syllables on one note, may be selected by setting the duration to `\breve`. The following example demonstrates the various Kievan note heads:

```
\autoBeamOff
\cadenzaOn
\kievanOn
b'1 b'2 b'4 b'8 b'\breve b'\longa
\kievanOff
b'2
```



Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*, Sezione “note head” in *Glossario Musicale*.

Notation Reference: Sezione A.9 [Note head styles], pagina 653.
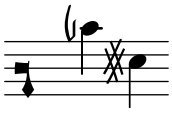
Problemi noti e avvertimenti

LilyPond automatically determines if the stem up or stem down form of a note is drawn. When setting chant in square notation, however, it is customary to have the stems point in the same direction within a single melisma. This can be done manually by setting the `direction` property of the `Stem` object.

Kievan accidentals

The `kievan` style for accidentals is selected with the `glyph-name-alist` property of the grob `Accidental`. The `kievan` style provides a sharp and a flat sign different from the default style. There is no natural sign in Kievan notation. The sharp sign is not used in Synodal music but may occur in earlier manuscripts. It has been included primarily for the sake of compatibility.

```
\clef "kievan-do"
\override Accidental.glyph-name-alist =
 #alteration-kievan-glyph-name-alist
bes' dis,
```



Vedi anche
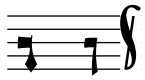
Music Glossary: Sezione “kievan notation” in *Glossario Musicale*, Sezione “accidental” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Accidentals], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic accidentals], pagina ⟨undefined⟩, Sezione A.8 [The Feta font], pagina 632

Kievan bar line

A decorative figure is commonly placed at the end of a piece of Kievan notation, which may be called the Kievan final bar line. It can be invoked as `\bar "k"`.

```
\kievanOn
\clef "kievan-do"
c \bar "k"
```



Vedi anche

⟨undefined⟩ [Bars], pagina ⟨undefined⟩, Sezione A.8 [The Feta font], pagina 632

Kievan melismata

Notes within a Kievan melisma are usually placed close to each other and the melismata separated by whitespace. This is done to allow the chanter to quickly identify the melodic structures of Znamenny chant. In LilyPond, melismata are treated as ligatures and the spacing is implemented by the `Kievan_ligature_engraver`.
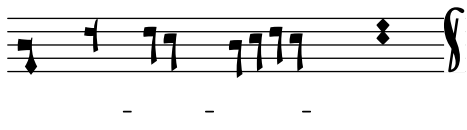
When the `KievanVoice` and `KievanStaff` contexts are used, the `Kievan_ligature_engraver` is enabled by default. In other contexts, it can be invoked by replacing the `Ligature_bracket_engraver` with the `Kievan_ligature_engraver` in the layout block:

```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Kievan_ligature_engraver"
  }
}
```

The spacing between the notes within a Kievan ligature can be controlled by setting the `padding` property of the `KievanLigature`.

The following example demonstrates the use of Kievan ligatures:

```
\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
        e2 \[ e4( d4 ) \] \[ c4( d e  d ) \] e1 \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Га -- вpi -- и -- лу
    }
  >>
}
```



Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [White mensural ligatures], pagina 423, [Gregorian square neume ligatures], pagina 428, [Ligatures], pagina 415.

Problemi noti e avvertimenti

Horizontal spacing of ligatures is poor.

2.9.6 Working with ancient music—scenarios and solutions

Working with ancient music frequently involves particular tasks which differ considerably from the modern notation for which LilyPond is designed. In the rest of this section, a number of typical scenarios are outlined, with suggestions of solutions. These involve:

- how to make incipits (i.e. prefatory material to indicate what the original has looked like) to modern transcriptions of mensural music;
- how to achieve the *Mensurstriche* layout frequently used for modern transcriptions of polyphonic music;
- how to transcribe Gregorian chant in modern notation;
- how to generate both ancient and modern notation from the same source.

Incipits

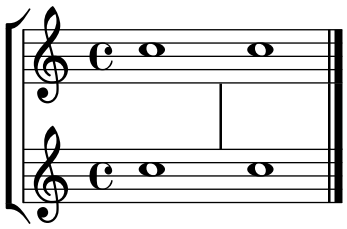
TBC

Mensurstriche layout

Mensurstriche ('mensuration lines') is the accepted term for bar lines that are drawn between the staves of a system but not through the staves themselves. It is a common way to preserve the rhythmic appearance of the original, i.e. not having to break syncopated notes at bar lines, while still providing the orientation aids that bar lines give.

La formattazione mensurale, in cui le stanghette non appaiono sui righi ma nello spazio tra i righi, si può ottenere usando `StaffGroup` al posto di `ChoirStaff`. La stanghetta sui righi viene nascosta impostando la proprietà `transparent`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Transcribing Gregorian chant

Gregorian chant can be transcribed into modern notation with a number of simple tweaks.

Stems. Stems can be left out altogether by `\remove`-ing the `Stem_engraver` from the Voice context:

```
\layout {
  ...
```

```
  \context {
    \Voice
      \remove "Stem_engraver"
  }
}
```

However, in some transcription styles, stems are used occasionally, for example to indicate the transition from a single-tone recitative to a fixed melodic gesture. In these cases, one can use either `\hide Stem` or `\override Stem.length = #0` instead, and restore the stem when needed with the corresponding `\once \override Stem.transparent = ##f` (see example below).

Timing. For unmetered chant, there are several alternatives.

The Time_signature_engraver can be removed from the Staff context without any negative side effects. The alternative, to make it transparent, will leave an empty space in the score, since the invisible signature will still take up space.

In many cases, `\set Score.timing = ##f` will give good results. Another alternative is to use `\cadenzaOn` and `\cadenzaOff`.

To remove the bar lines, the radical approach is to `\remove` the Bar_engraver from the Staff context. Again, one may want to use `\hide BarLine` instead, if an occasional barline is wanted.

A common type of transcription is recitativic chant where the repeated notes are indicated with a single breve. The text to the recitation tone can be dealt with in two different ways: either set as a single, left-aligned syllable:

```
\include "gregorian.ly"
chant = \relative c' {
  \clef "G_8"
  c\breve c4 b4 a c2 c4  \divisioMaior
  c\breve c4 c f, f \finalis
}

verba = \lyricmode {
  \once \override LyricText.self-alignment-X = #-1
  "Noctem quietam et" fi -- nem per -- fec -- tum
  \once \override LyricText.self-alignment-X = #-1
  "concedat nobis Dominus" om -- ni -- po -- tens.
}
\score {
  \new Staff <<
  \new Voice = "melody" \chant
  \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
  }
}
```



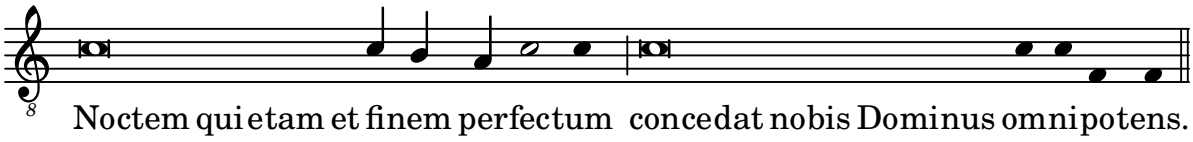


This works fine, as long as the text doesn't span a line break. If that is the case, an alternative is to add hidden notes to the score, here in combination with changing stem visibility:

```
\include "gregorian.ly"
chant = \relative c' {
  \clef "G_8"
  \set Score.timing = ##f
  c\breve \hide NoteHead  c c c c c
  \undo \hide NoteHead
  \override Stem.transparent = ##f \stemUp c4 b4 a
  \hide Stem c2 c4  \divisioMaior
  c\breve \hide NoteHead c c c c c c c
  \undo \hide NoteHead c4 c f, f \finalis
}

verba = \lyricmode {
  No -- ctem qui -- e -- tam et fi -- nem per -- fec -- tum
  con -- ce -- dat no -- bis Do -- mi -- nus om -- ni -- po -- tens.
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics \lyricsto "melody" \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \hide BarLine
      \hide Stem
    }
  }
}
```
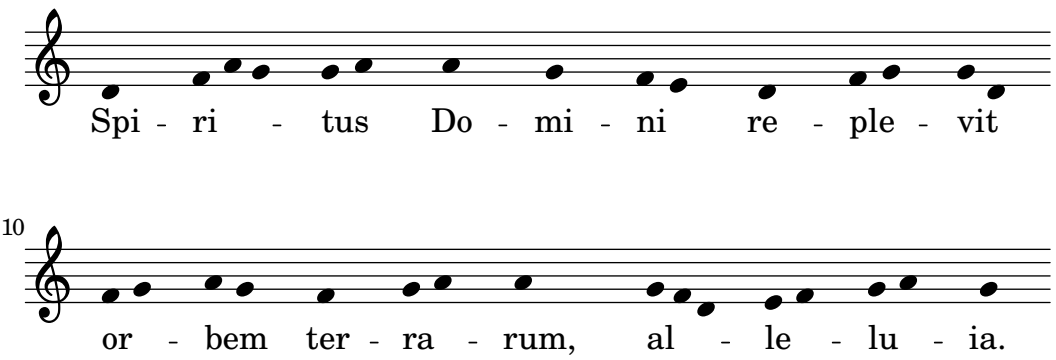


Another common situation is transcription of neumatic or melismatic chants, i.e. chants with a varying number of notes to each syllable. In this case, one would want to set the syllable groups clearly apart, usually also the subdivisions of a longer melisma. One way to achieve this is to use a fixed `\time`, e.g., 1/4, and let each syllable or note group fill one of these measures, with the help of tuplets or shorter durations. If the bar lines and all other rhythmical indications are made transparent, and the space around the bar lines is increased, this will give a fairly good representation in modern notation of the original.

To avoid that syllables of different width (such as “-ri” and “-rum”) spread the syllable note groups unevenly apart, the `'X-extent` property of the `LyricText` object may be set to a fixed value. Another, more cumbersome way would be to add the syllables as `\markup` elements. If further adjustments are necessary, this can be easily done with `s` ‘notes’.

```
spiritus = \relative c' {
  \time 1/4
  \override Lyrics.LyricText.X-extent  = #'(0 . 3)
  d4 \tuplet 3/2 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4  s
  \tuplet 3/2 { g8 f d } e f g a g4
}

spirLyr = \lyricmode {
  Spi -- ri -- _ _ tus  _ Do -- mi -- ni  _ re -- ple -- _ vit _
  or -- _ bem _  ter -- ra -- _ rum, al -- _ _ le -- _ lu
  -- _ ia.
}
\score {
  \new Staff <<
    \new Voice = "chant" \spiritus
    \new Lyrics = "one" \lyricsto "chant" \spirLyr
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \override BarLine.X-extent = #'(-1 . 1)
      \hide Stem
      \hide Beam
      \hide BarLine
      \hide TupletNumber
    }
  }
}
```



Ancient and modern from one source

TBC

Editorial markings

2.10 World music

The purpose of this section is to highlight musical notation issues that are relevant to traditions outside the Western tradition.

2.10.1 Common notation for non-Western music

This section discusses how to enter and print music scores that do not belong to the Western classical tradition, also referred to as *Common Practice Period*.

Extending notation and tuning systems

Standard classical notation (also known as *Common Practice Period* notation) is commonly used in all sorts of music, not limited to 'classical' Western music. This notation is discussed in ⟨undefined⟩ [Writing pitches], pagina ⟨undefined⟩, and the various note names that may be used are explained in ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩.

However, many types of non-Western music (and some types of Western folk and traditional music) employ alternative or extended tuning systems that do not fit readily into standard classical notation.

In some cases standard notation is still used, with the pitch differences being implicit. For example, *Arabic music* is notated with standard semitone and quarter-tone accidentals, with the precise pitch alterations being determined by context. Italian note names are typically used, while the init file '`arabic.ly`' provides a suitable set of macros and definitions extending the standard notation. For more details, see Sezione 2.10.2 [Arabic music], pagina 444.

Other types of music require extended or unique notations. *Turkish classical music* or Ottoman music, for example, employs melodic forms known as *makamlar*, whose intervals are based on 1/9 divisions of the whole tone. Standard Western staff notes are still used, but with special accidentals unique to Turkish music, that are defined in the file '`makam.ly`'. For further information on Turkish classical music and makamlar, see Sezione 2.10.3 [Turkish classical music], pagina 449.

To locate init files such as '`arabic.ly`' or '`makam.ly`' on your system, see Sezione "Other sources of information" in *Manuale di Apprendimento*.
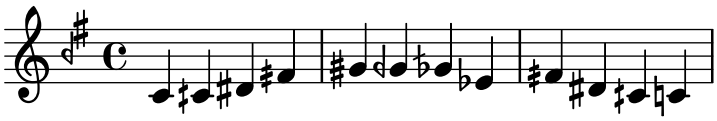
Frammenti di codice selezionati

Makam example

Makam is a type of melody from Turkey using 1/9th-tone microtonal alterations. Consult the initialization file '`ly/makam.ly`' for details of pitch names and alterations.

```
% Initialize makam settings
\include "makam.ly"

\relative c' {
  \set Staff.keySignature = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```



Vedi anche

Music Glossary: Sezione “Common Practice Period” in *Glossario Musicale*, Sezione “makamlar” in *Glossario Musicale*.

Learning Manual: Sezione “Other sources of information” in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Writing pitches], pagina ⟨undefined⟩, ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩, Sezione 2.10.2 [Arabic music], pagina 444, Sezione 2.10.3 [Turkish classical music], pagina 449.

2.10.2 Arabic music

This section highlights issues that are relevant to notating Arabic music.

References for Arabic music

Arabic music so far has been mainly an oral tradition. When music is transcribed, it is usually in a sketch format, on which performers are expected to improvise significantly. Increasingly, Western notation, with a few variations, is adopted in order to communicate and preserve Arabic music.

Some elements of Western musical notation such as the transcription of chords or independent parts, are not required to typeset the more traditional Arabic pieces. There are however some different issues, such as the need to indicate medium intervals that are somewhere between a semi-tone and a tone, in addition to the minor and major intervals that are used in Western music. There is also the need to group and indicate a large number of different maqams (modes) that are part of Arabic music.

In general, Arabic music notation does not attempt to precisely indicate microtonal elements that are present in musical practice.

Several issues that are relevant to Arabic music are covered elsewhere:

- Note names and accidentals (including quarter tones) can be tailored as discussed in Sezione 2.10.1 [Common notation for non-Western music], pagina 443.
- Additional key signatures can also be tailored as described in ⟨undefined⟩ [Key signature], pagina ⟨undefined⟩.
- Complex time signatures may require that notes be grouped manually as described in ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩.
- *Takasim* which are rhythmically free improvisations may be written down omitting bar lines as described in ⟨undefined⟩ [Unmetered music], pagina ⟨undefined⟩.

Vedi anche

Notation Reference: Sezione 2.10.1 [Common notation for non-Western music], pagina 443, ⟨undefined⟩ [Key signature], pagina ⟨undefined⟩, ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩.
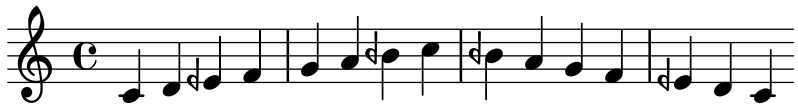
Snippets: Sezione “World music” in *Frammenti di codice*.

Arabic note names

The more traditional Arabic note names can be quite long and are not suitable for the purpose of music writing, so they are not used. English note names are not very familiar in Arabic music education, so Italian or Solfege note names (`do, re, mi, fa, sol, la, si`) are used instead; modifiers (accidentals) can also be used. Italian note names and accidentals are explained in ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩; the use of standard Western notation to notate non-Western music is discussed in Sezione 2.10.1 [Common notation for non-Western music], pagina 443.
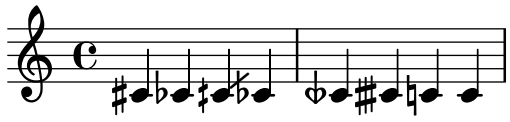
For example, this is how the Arabic *rast* scale can be notated:

```
\include "arabic.ly"
\relative do' {
  do re misb fa sol la sisb do sisb la sol fa misb re do
}
```



The symbol for semi-flat does not match the symbol which is used in Arabic notation. The `\dwn` symbol defined in '`arabic.ly`' may be used preceding a flat symbol as a work around if it is important to use the specific Arabic semi-flat symbol. The appearance of the semi-flat symbol in the key signature cannot be altered by using this method.

```
\include "arabic.ly"
\relative do' {
  \set Staff.extraNatural = ##f
  dod dob dosd \dwn dob dobsb dodsd do do
}
```



Vedi anche

Notation Reference: ⟨undefined⟩ [Note names in other languages], pagina ⟨undefined⟩, Sezione 2.10.1 [Common notation for non-Western music], pagina 443.

Snippets: Sezione "World music" in *Frammenti di codice*.

Arabic key signatures

In addition to the minor and major key signatures, the following key signatures are defined in '`arabic.ly`': *bayati*, *rast*, *sikah*, *iraq*, and *kurd*. These key signatures define a small number of maqam groups rather than the large number of maqams that are in common use.

In general, a maqam uses the key signature of its group, or a neighbouring group, and varying accidentals are marked throughout the music.

For example to indicate the key signature of a maqam muhayer piece:

```
\key re \bayati
```

Here *re* is the default pitch of the muhayer maqam, and *bayati* is the name of the base maqam in the group.

While the key signature indicates the group, it is common for the title to indicate the more specific maqam, so in this example, the name of maqam muhayer should appear in the title.

Other maqams in the same bayati group, as shown in the table below: (bayati, hussaini, saba, and ushaq) can be indicated in the same way. These are all variations of the base and most common maqam in the group, which is bayati. They usually differ from the base maqam in their upper tetrachords, or certain flow details that don't change their fundamental nature, as siblings.

The other maqam in the same group (Nawa) is related to bayati by modulation which is indicated in the table in parenthesis for those maqams that are modulations of their base maqam. Arabic maqams admit of only limited modulations, due to the nature of Arabic musical instruments. Nawa can be indicated as follows:

`\key sol \bayati`

In Arabic music, the same term such as bayati that is used to indicate a maqam group, is also a maqam which is usually the most important in the group, and can also be thought of as a base maqam.

Here is one suggested grouping that maps the more common maqams to key signatures:

maqam group	**key**	**finalis**	**Other maqmas in group (finalis)**
ajam	major	sib	jaharka (fa)
bayati	bayati	re	hussaini, muhayer, saba, ushaq, nawa (sol)
hijaz	kurd	re	shahnaz, shad arban (sol), hijazkar (do)
iraq	iraq	sisb	-
kurd	kurd	re	hijazkar kurd (do)
nahawand	minor	do	busalik (re), farah faza (sol)
nakriz	minor	do	nawa athar, hisar (re)
rast	rast	do	mahur, yakah (sol)
sikah	sikah	misb	huzam

Frammenti di codice selezionati

Armature di chiave non tradizionali

Il comando `\key` comunemente usato imposta la proprietà `keySignature`, che fa parte del contesto `Staff`.
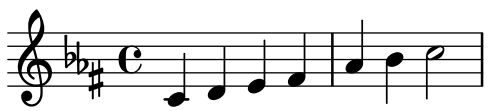
Per creare armature di chiave non standard, tale proprietà va impostata esplicitamente. Il formato di questo comando è una lista:

`\set Staff.keySignature = #`(((ottava . grado) . alterazione) ((ottava . grado) . alterazione) ...)` dove, per ogni elemento della lista, `ottava` indica l'ottava (0 è l'ottava dal Do centrale al Si precedente), `grado` indica la nota all'interno dell'ottava (0 significa Do e 6 significa Si) e `alterazione` può essere `,SHARP` `,FLAT` `,DOUBLE-SHARP` etc. (Si noti la virgola iniziale.)

Altrimenti, usando, per ogni elemento della lista, il formato breve `(grado . alterazione)`, ciò indica che la stessa alterazione deve essere presente in tutte le ottave.

Ecco un esempio di una possibile armatura per generare una scala a tono intero:

```
\relative c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
                              ((0 . 5) . ,FLAT)
                              ((0 . 3) . ,SHARP))
  c4 d e fis
  aes4 bes c2
}
```



Vedi anche

Music Glossary: Sezione “maqam” in *Glossario Musicale*, Sezione “bayati” in *Glossario Musicale*, Sezione “rast” in *Glossario Musicale*, Sezione “sikah” in *Glossario Musicale*, Sezione “iraq” in *Glossario Musicale*, Sezione “kurd” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Key signature], pagina ⟨undefined⟩.

Learning Manual: Sezione “Accidentals and key signatures” in *Manuale di Apprendimento*.

Internals Reference: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

Snippets: Sezione “World music” in *Frammenti di codice*, Sezione “Pitches” in *Frammenti di codice*.

Arabic time signatures

Some Arabic and Turkish music classical forms such as *Semai* use unusual time signatures such as 10/8. This may lead to an automatic grouping of notes that is quite different from existing typeset music, where notes may not be grouped on the beat, but in a manner that is difficult to match by adjusting automatic beaming. The alternative is to switch off automatic beaming and beam the notes manually. Even if a match to existing typeset music is not required, it may still be desirable to adjust the automatic beaming behaviour and/or use compound time signatures.

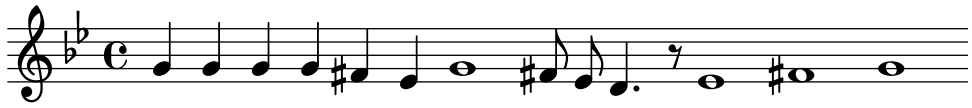
Frammenti di codice selezionati

Arabic improvisation

For improvisations or taqasim which are temporarily free, the time signature can be omitted and `\cadenzaOn` can be used. Adjusting the accidental style might be required, since the absence of bar lines will cause the accidental to be marked only once. Here is an example of what could be the start of a hijaz improvisation:

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



Vedi anche

Music Glossary: Sezione “semai” in *Glossario Musicale*, Sezione “taqasim” in *Glossario Musicale*.

Notation Reference: ⟨undefined⟩ [Manual beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic beams], pagina ⟨undefined⟩, ⟨undefined⟩ [Unmetered music], pagina ⟨undefined⟩, ⟨undefined⟩ [Automatic accidentals], pagina ⟨undefined⟩, ⟨undefined⟩ [Setting automatic beam behavior], pagina ⟨undefined⟩, ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.

Snippets: Sezione “World music” in *Frammenti di codice*.
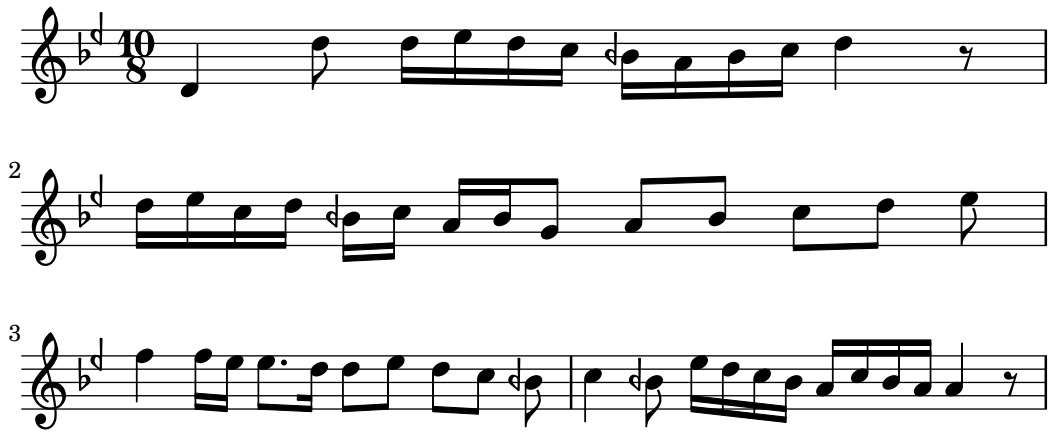
Arabic music example

Here is a template that also uses the start of a Turkish *Semai* that is familiar in Arabic music education in order to illustrate some of the peculiarities of Arabic music notation, such as medium intervals and unusual modes that are discussed in this section.

```
\include "arabic.ly"
\score {
  \relative re' {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
```

```
    \key re \bayati
    \time 10/8

    re4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
    re16 [misb do re] sisb [do] la [sisb sol8] la [sisb] do [re] misb
    fa4 fa16 [misb] misb8. [re16] re8 [misb] re  [do] sisb
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
  }
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
}
```



Vedi anche

Snippets: Sezione “World music” in *Frammenti di codice*.

Further reading for Arabic music

1. *The music of the Arabs* by Habib Hassan Touma [Amadeus Press, 1996], contains a discussion of maqams and their method of groupings.

   There are also various web sites that explain maqams and some provide audio examples such as :

   - http://www.maqamworld.com/
   - http://www.turath.org/

   There are some variations in the details of how maqams are grouped, despite agreement on the criteria of grouping maqams that are related through common lower tetra chords, or through modulation.

2. There is not a complete consistency, sometimes even in the same text on how key signatures for particular maqams should be specified. It is common, however, to use a key signature per group, rather than a different key signature for each different maqam.

   Method books by the following authors for the *Oud*, the Arabic lute, contain examples of mainly Turkish and Arabic compositions.

   - Charbel Rouhana
   - George Farah
   - Ibrahim Ali Darwish Al-masri

2.10.3 Turkish classical music

This section highlights issues that are relevant to notating Turkish classical music.

References for Turkish classical music

Turkish classical music developed in the Ottoman Empire in a period roughly contemporaneous with classical music in Europe, and has continued on into the 20th and 21st centuries as a vibrant and distinct tradition with its own compositional forms, theory and performance styles. Among its striking features is the use of microtonal intervals based on 'commas' of 1/9 of a tone, from which are constructed the melodic forms known as *makam* (plural *makamlar*).

Some issues relevant to Turkish classical music are covered elsewhere:

- Special note names and accidentals are explained in Sezione 2.10.1 [Common notation for non-Western music], pagina 443.

Turkish note names

Pitches in Turkish classical music traditionally have unique names, and the basis of pitch on 1/9-tone divisions means makamlar employ a completely different set of intervals from Western scales and modes: *koma* (1/9 of a tone), *eksik bakiye* (3/9), *bakiye* (4/9), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanîni* (a whole tone) and *artık ikili* (12/9 or 13/9 of a tone).

From a modern notational point of view it is convenient to use the standard Western staff notes (c, d, e, ...) with special accidentals that raise or lower notes by intervals of 1/9, 4/9, 5/9 and 8/9 of a tone. These accidentals are defined in the file '`makam.ly`'.

The following table lists:

- the name of these special accidentals,
- the accidental suffix that must be added to notes,
- and their pitch alteration as a fraction of one whole tone.

Accidental name	**suffix**	**pitch alteration**
büyük mücenneb (sharp)	-bm	+8/9
kücük mücenneb (sharp)	-k	+5/9
bakiye (sharp)	-b	+4/9
koma (sharp)	-c	+1/9
koma (flat)	-fc	-1/9
bakiye (flat)	-fb	-4/9
kücük mücenneb (flat)	-fk	-5/9
büyük mücenneb (flat)	-fbm	-8/9

For a more general explanation of non-Western music notation, see Sezione 2.10.1 [Common notation for non-Western music], pagina 443.

Vedi anche

Music Glossary: Sezione "makam" in *Glossario Musicale*, Sezione "makamlar" in *Glossario Musicale*.

Notation Reference: Sezione 2.10.1 [Common notation for non-Western music], pagina 443.

3 General input and output

This section deals with general LilyPond input and output issues, rather than specific notation.

3.1 Input structure

The main format of input for LilyPond are text files. By convention, these files end with '`.ly`'.

3.1.1 Structure of a score

A `\score` block must contain a single music expression delimited by curly brackets:
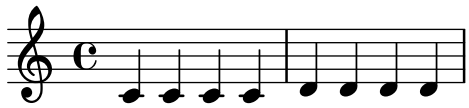
```
\score {
  ...
}
```

> **Nota:** There must be **only one** outer music expression in a `\score` block, and it **must** be surrounded by curly brackets.

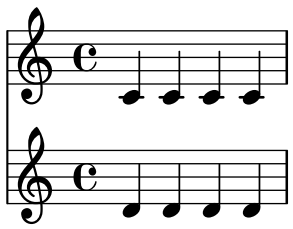
This single music expression may be of any size, and may contain other music expressions to any complexity. All of these examples are music expressions:

```
{ c'4 c' c' c' }
```

```
{
  { c'4 c' c' c' }
  { d'4 d' d' d' }
}
```



```
<<
  \new Staff { c'4 c' c' c' }
  \new Staff { d'4 d' d' d' }
>>
```



```
{
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { \flute }
      \new Staff { \oboe }
    >>
    \new StaffGroup <<
      \new Staff { \violinI }
      \new Staff { \violinII }
    >>
  >>
```

```
}
```

Comments are one exception to this general rule. (For others see Sezione 3.1.5 [File structure], pagina 454.) Both single-line comments and comments delimited by `%{ ... %}` may be placed anywhere within an input file. They may be placed inside or outside a `\score` block, and inside or outside the single music expression within a `\score` block.

Remember that even in a file containing only a `\score` block, it is implicitly enclosed in a \book block. A \book block in a source file produces at least one output file, and by default the name of the output file produced is derived from the name of the input file, so '`fandangoforelephants.ly`' will produce '`fandangoforelephants.pdf`'.

(For more details about `\book` blocks, see Sezione 3.1.2 [Multiple scores in a book], pagina 451, Sezione 3.1.3 [Multiple output files from one input file], pagina 452 Sezione 3.1.5 [File structure], pagina 454.)

Vedi anche

Learning Manual: Sezione "Working on input files" in *Manuale di Apprendimento*, Sezione "Music expressions explained" in *Manuale di Apprendimento*, Sezione "Score is a (single) compound musical expression" in *Manuale di Apprendimento*.

3.1.2 Multiple scores in a book

A document may contain multiple pieces of music and text. Examples of these are an etude book, or an orchestral part with multiple movements. Each movement is entered with a `\score` block,

```
\score {
  ...music...
}
```

and texts are entered with a `\markup` block,

```
\markup {
  ...text...
}
```

All the movements and texts which appear in the same '`.ly`' file will normally be typeset in the form of a single output file.

```
\score {
  ...
}
\markup {
  ...
}
\score {
  ...
}
```

One important exception is within lilypond-book documents, where you explicitly have to add a `\book` block, otherwise only the first `\score` or `\markup` will appear in the output.

The header for each piece of music can be put inside the `\score` block. The `piece` name from the header will be printed before each movement. The title for the entire book can be put inside the `\book`, but if it is not present, the `\header` which is at the top of the file is inserted.

```
\header {
  title = "Eight miniatures"
  composer = "Igor Stravinsky"
}
```

```
\score {
  ...
  \header { piece = "Romanze" }
}
\markup {
   ...text of second verse...
}
\markup {
   ...text of third verse...
}
\score {
  ...
  \header { piece = "Menuetto" }
}
```

Pieces of music may be grouped into book parts using `\bookpart` blocks. Book parts are separated by a page break, and can start with a title, like the book itself, by specifying a `\header` block.

```
\bookpart {
  \header {
    title = "Book title"
    subtitle = "First part"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Second part"
  }
  \score { ... }
  ...
}
```

3.1.3 Multiple output files from one input file

If you want multiple output files from the same '`.ly`' file, then you can add multiple `\book` blocks, where each such \book block will result in a separate output file. If you do not specify any `\book` block in the input file, LilyPond will implicitly treat the whole file as a single \book block, see Sezione 3.1.5 [File structure], pagina 454.

When producing multiple files from a single source file, Lilypond ensures that none of the output files from any `\book` block overwrites the output file produced by a preceding `\book` from the same input file.

It does this by adding a suffix to the output name for each `\book` which uses the default output file name derived from the input source file.

The default behaviour is to append a version-number suffix for each name which may clash, so

```
\book {
  \score { ... }
  \paper { ... }
}
\book {
```

```
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
```

in source file ‘`eightminiatures.ly`’ will produce

- ‘`eightminiatures.pdf`’,
- ‘`eightminiatures-1.pdf`’ and
- ‘`eightminiatures-2.pdf`’.

3.1.4 Output file names

Lilypond provides facilities to allow you to control what file names are used by the various back-ends when producing output files.

In the previous section, we saw how Lilypond prevents name-clashes when producing several ouputs from a single source file. You also have the ability to specify your own suffixes for each `\book` block, so for example you can produce files called ‘`eightminiatures-Romanze.pdf`’, ‘`eightminiatures-Menuetto.pdf`’ and ‘`eightminiatures-Nocturne.pdf`’ by adding a `\bookOutputSuffix` declaration inside each `\book` block.

```
\book {
  \bookOutputSuffix "Romanze"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Menuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Nocturne"
  \score { ... }
  \paper { ... }
}
```

You can also specify a different output filename for `book` block, by using `\bookOutputName` declarations

```
\book {
  \bookOutputName "Romanze"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Menuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Nocturne"
  \score { ... }
```

```
  \paper { ... }
}
```

The file above will produce these output files:

- ‘Romanze.pdf’,
- ‘Menuetto.pdf’ and
- ‘Nocturne.pdf’.

3.1.5 File structure

A ‘.ly’ file may contain any number of toplevel expressions, where a toplevel expression is one of the following:

- An output definition, such as \paper, \midi, and \layout. Such a definition at the toplevel changes the default book-wide settings. If more than one such definition of the same type is entered at the top level the definitions are combined, but in conflicting situations the later definitions take precedence. For details of how this affects the \layout block see Sezione 4.2.1 [The \layout block], pagina 512.
- A direct scheme expression, such as `#(set-default-paper-size "a7" 'landscape)` or `#(ly:set-option 'point-and-click #f)`.
- A \header block. This sets the global (i.e. the top of file) header block. This is the block containing the default settings of titling fields like composer, title, etc. for all books within the file (see [Titles explained], pagina 456).
- A \score block. This score will be collected with other toplevel scores, and combined as a single \book. This behavior can be changed by setting the variable toplevel-score-handler at toplevel. The default handler is defined in the init file ‘../scm/lily.scm’.
- A \book block logically combines multiple movements (i.e., multiple \score blocks) in one document. If there are a number of \scores, one output file will be created for each \book block, in which all corresponding movements are concatenated. The only reason to explicitly specify \book blocks in a ‘.ly’ file is if you wish to create multiple output files from a single input file. One exception is within lilypond-book documents, where you explicitly have to add a \book block if you want more than a single \score or \markup in the same example. This behavior can be changed by setting the variable toplevel-book-handler at toplevel. The default handler is defined in the init file ‘../scm/lily.scm’.
- A \bookpart block. A book may be divided into several parts, using \bookpart blocks, in order to ease the page breaking, or to use different \paper settings in different parts.
- A compound music expression, such as

  ```
  { c'4 d' e'2 }
  ```

  This will add the piece in a \score and format it in a single book together with all other toplevel \scores and music expressions. In other words, a file containing only the above music expression will be translated into

  ```
  \book {
    \score {
      \new Staff {
        \new Voice {
          { c'4 d' e'2 }
        }
      }
      \layout { }
    }
    \paper { }
  ```

```
  \header { }
}
```

This behavior can be changed by setting the variable `toplevel-music-handler` at toplevel. The default handler is defined in the init file '`../scm/lily.scm`'.

- A markup text, a verse for example

  ```
  \markup {
     2.  The first line verse two.
  }
  ```

  Markup texts are rendered above, between or below the scores or music expressions, wherever they appear.

- A variable, such as

  ```
  foo = { c4 d e d }
  ```

  This can be used later on in the file by entering `\foo`. The name of a variable should have alphabetic characters only; no numbers, underscores or dashes.

The following example shows three things that may be entered at toplevel

```
\layout {
  % Don't justify the output
  ragged-right = ##t
}

\header {
   title = "Do-re-mi"
}

{ c'4 d' e2 }
```

At any point in a file, any of the following lexical instructions can be entered:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- A single-line comment, introduced by a leading `%` sign.
- A multi-line comment delimited by `%{ ... %}`.

Whitespace between items in the input stream is generally ignored, and may be freely omitted or extended to enhance readability. However, whitespace should always be used in the following circumstances to avoid errors:

- Around every opening and closing curly bracket.
- After every command or variable, i.e. every item that begins with a `\` sign.
- After every item that is to be interpreted as a Scheme expression, i.e. every item that begins with a `#` sign.
- To separate all elements of a Scheme expression.
- In `lyricmode` before and after `\set` and `\override` commands.

Vedi anche

Learning Manual: Sezione "How LilyPond input files work" in *Manuale di Apprendimento*.

Notation Reference: [Titles explained], pagina 456, Sezione 4.2.1 [The `\layout` block], pagina 512.

3.2 Titles and headers

Almost all printed music includes a title and the composer's name; some pieces include a lot more information.

3.2.1 Creating titles headers and footers

Titles explained

Each `\book` block in a single input file produces a separate output file, see Sezione 3.1.5 [File structure], pagina 454. Within each output file three types of titling areas are provided: *Book Titles* at the beginning of each book, *Bookpart Titles* at the beginning of each bookpart and *Score Titles* at the beginning of each score.

Values of titling fields such as `title` and `composer` are set in `\header` blocks. (For the syntax of `\header` blocks and a complete list of the fields available by default see [Default layout of bookpart and score titles], pagina 459). Book Titles, Bookpart Titles and Score Titles can all contain the same fields, although by default the fields in Score Titles are limited to `piece` and `opus`.

`\header` blocks may be placed in four different places to form a descending hierarchy of `\header` blocks:

- At the top of the input file, before all `\book`, `\bookpart`, and `\score` blocks.
- Within a `\book` block but outside all the `\bookpart` and `\score` blocks within that book.
- Within a `\bookpart` block but outside all `\score` blocks within that bookpart.
- After the music expression in a `\score` block.

The values of the fields filter down this hierarchy, with the values set higher in the hierarchy persisting unless they are over-ridden by a value set lower in the hierarchy, so:

- A Book Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block. The resulting fields are used to print the Book Title for that book, providing that there is other material which generates a page at the start of the book, before the first bookpart. A single `\pageBreak` will suffice.
- A Bookpart Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block, and further modified by fields set in the `\bookpart` block. The resulting values are used to print the Bookpart Title for that bookpart.
- A Score Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block, further modified by fields set in the `\bookpart` block and finally modified by fields set in the `\score` block. The resulting values are used to print the Score Title for that score. Note, though, that only `piece` and `opus` fields are printed by default in Score Titles unless the `\paper` variable, `print-all-headers`, is set to `#t`.

Nota: Remember when placing a `\header` block inside a `\score` block, that the music expression must come before the `\header` block.

It is not necessary to provide `\header` blocks in all four places: any or even all of them may be omitted. Similarly, simple input files may omit the `\book` and `\bookpart` blocks, leaving them to be created implicitly.

If the book has only a single score, the `\header` block should normally be placed at the top of the file so that just a Bookpart Title is produced, making all the titling fields available for use.

If the book has multiple scores a number of different arrangements of `\header` blocks are possible, corresponding to the various types of musical publications. For example, if the publication contains several pieces by the same composer a `\header` block placed at the top of the file

specifying the book title and the composer with `\header` blocks in each `\score` block specifying the `piece` and/or `opus` would be most suitable, as here:

```
\header {
  title = "SUITE I."
  composer = "J. S. Bach."
}
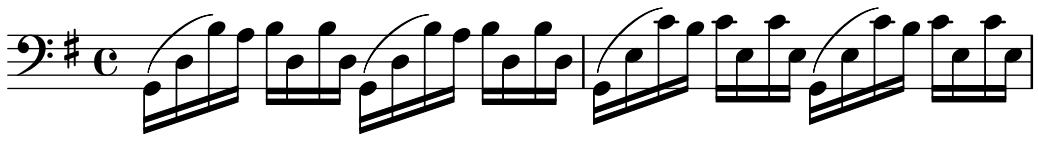
\score {
  \new Staff \relative g, {
    \clef bass
    \key g \major
    \repeat unfold 2 { g16( d' b') a b d, b' d, } |
    \repeat unfold 2 { g,16( e' c') b c e, c' e, } |
  }
  \header {
    piece = "Prélude."
  }
}

\score {
  \new Staff \relative b {
    \clef bass
    \key g \major
    \partial 16 b16 |
    <g, d' b'~>4 b'16 a( g fis) g( d e fis) g( a b c) |
    d16( b g fis) g( e d c) b(c d e) fis( g a b) |
  }
  \header {
    piece = "Allemande."
  }
}
```
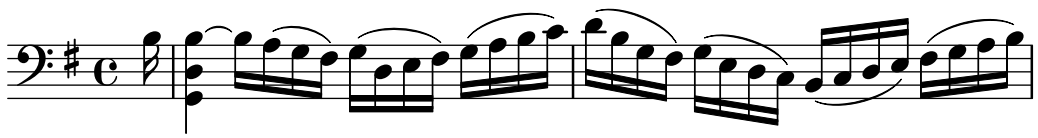
SUITE I.

J. S. Bach.

Prélude.



Allemande.



More complicated arrangements are possible. For example, text fields from the `\header` block in a book can be displayed in all Score Titles, with some fields over-ridden and some manually suppressed:

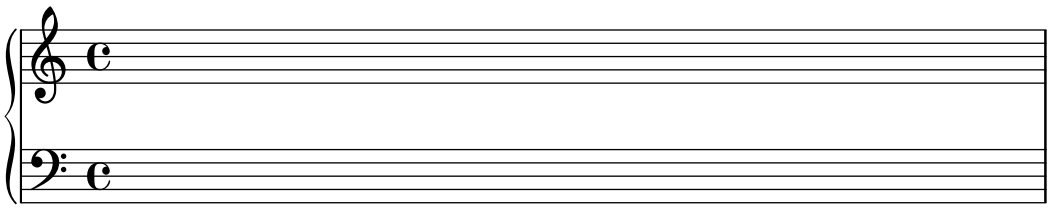
```
\book {
  \paper {
    print-all-headers = ##t
  }
  \header {
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    % Do not display the tagline for this book
    tagline = ##f
  }
  \markup { \vspace #1 }
  \score {
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
    \header {
      title = "PRAELUDIUM I"
      opus = "BWV 846"
      % Do not display the subtitle for this score
      subtitle = ##f
    }
  }
  \score {
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
    \header {
      title = "FUGA I"
      subsubtitle = "A 4 VOCI"
      opus = "BWV 846"
      % Do not display the subtitle for this score
      subtitle = ##f
    }
  }
}
```

DAS WOHLTEMPERIRTE CLAVIER

TEIL I

PRAELUDIUM I

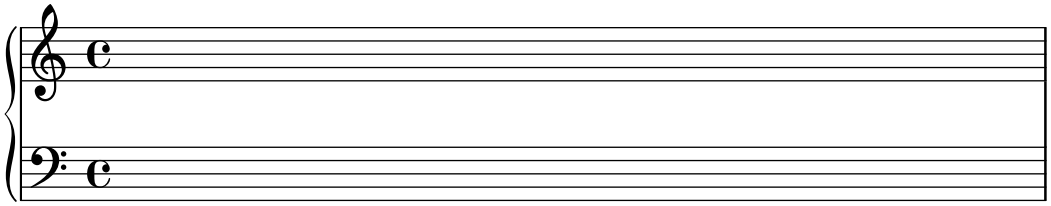
BWV 846



FUGA I

A 4 VOCI

BWV 846



Vedi anche

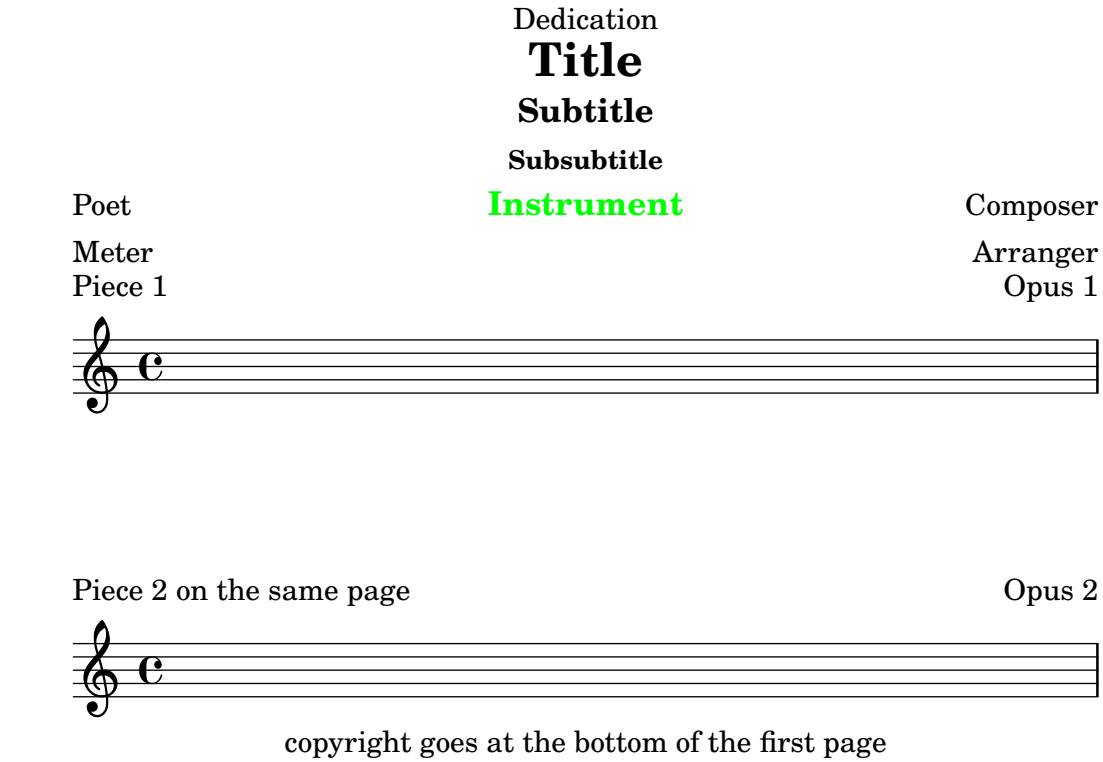
Notation Reference: Sezione 3.1.5 [File structure], pagina 454, [Default layout of bookpart and score titles], pagina 459, [Custom layout for titles], pagina 464.

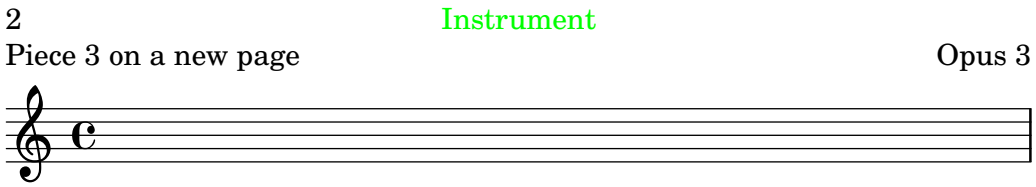
Default layout of bookpart and score titles

This example demonstrates all `\header` variables:

```
\book {
  \header {
      % The following fields are centered
    dedication = "Dedication"
    title = "Title"
    subtitle = "Subtitle"
    subsubtitle = "Subsubtitle"
      % The following fields are evenly spread on one line
      % the field "instrument" also appears on following pages
    instrument = \markup \with-color #green "Instrument"
    poet = "Poet"
    composer = "Composer"
      % The following fields are placed at opposite ends of the same line
    meter = "Meter"
    arranger = "Arranger"
      % The following fields are centered at the bottom
    tagline = "tagline goes at the bottom of the last page"
    copyright = "copyright goes at the bottom of the first page"
  }
  \score {
```

```
    { s1 }
    \header {
        % The following fields are placed at opposite ends of the same line
      piece = "Piece 1"
      opus = "Opus 1"
    }
  }
  \score {
    { s1 }
    \header {
        % The following fields are placed at opposite ends of the same line
      piece = "Piece 2 on the same page"
      opus = "Opus 2"
    }
  }
  \pageBreak
  \score {
    { s1 }
    \header {
        % The following fields are placed at opposite ends of the same line
      piece = "Piece 3 on a new page"
      opus = "Opus 3"
    }
  }
}
```





tagline goes at the bottom of the last page

Note that

- The instrument name will be repeated on every page.
- Only `piece` and `opus` are printed in a `\score` when the paper variable `print-all-headers` is set to `##f` (the default).
- Text fields left unset in a `\header` block are replaced with `\null` markups so that the space is not wasted.
- The default settings for `scoreTitleMarkup` place the `piece` and `opus` text fields at opposite ends of the same line.

To change the default layout see [Custom layout for titles], pagina 464.

If a `\book` block starts immediately with a `\bookpart` block, no Book Title will be printed, as there is no page on which to print it. If a Book Title is required, begin the `\book` block with some markup material or a `\pageBreak` command.

Use the `breakbefore` variable inside a `\header` block that is itself in a `\score` block, to make the higher-level `\header` block titles appear on the first page on their own, with the music (defined in the `\score` block) starting on the next.
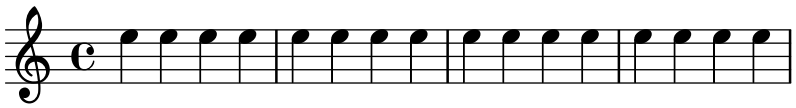
```
\book {
  \header {
    title = "This is my Title"
    subtitle = "This is my Subtitle"
    copyright = "This is the bottom of the first page"
  }
  \score {
    \repeat unfold 4 { e'' e'' e'' e'' }
    \header {
      piece = "This is the Music"
      breakbefore = ##t
    }
  }
}
```

This is my Title

This is my Subtitle

This is the bottom of the first page

2
This is the Music



Music engraving by LilyPond 2.18.2—www.lilypond.org

Vedi anche

Learning Manual: Sezione "How LilyPond input files work" in *Manuale di Apprendimento*,

Notation Reference: [Custom layout for titles], pagina 464, Sezione 3.1.5 [File structure], pagina 454.

Installed Files: '`ly/titling-init.ly`'.

Default layout of headers and footers

Headers and *footers* are lines of text appearing at the top and bottom of pages, separate from the main text of a book. They are controlled by the following `\paper` variables:
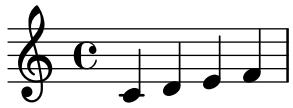
- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

These markup variables can only access text fields from top-level `\header` blocks (which apply to all scores in the book) and are defined in '`ly/titling-init.ly`'. By default:

- page numbers are automatically placed on the top far left (if even) or top far right (if odd), starting from the second page.
- the `instrument` text field is placed in the center of every page, starting from the second page.
- the `copyright` text is centered on the bottom of the first page.
- the `tagline` is centered on the bottom of the last page, and below the `copyright` text if there is only a single page.

The default tagline can be changed by adding a `tagline` in the top-level `\header` block.

```
\book {
  \header {
    tagline = "... music notation for Everyone"
  }
  \score {
    \relative c' {
      c4 d e f
    }
  }
}
```



... music notation for Everyone

To remove the `tagline` set the value to `##f`.

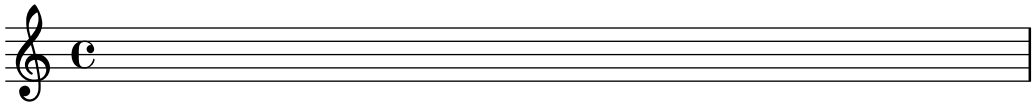
3.2.2 Custom titles headers and footers

Custom text formatting for titles

Standard `\markup` commands can be used to customize any header, footer and title text within the `\header` block.

```
\score {
  { s1 }
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
}
```

PRAELUDIUM I *BWV 846*



Vedi anche

Notation Reference: ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Custom layout for titles

`\markup` commands in the `\header` block are useful for simple text formatting, but they do not allow precise control over the placement of titles. To customize the placement of the text fields, change either or both of the following `\paper` variables:

- `bookTitleMarkup`
- `scoreTitleMarkup`

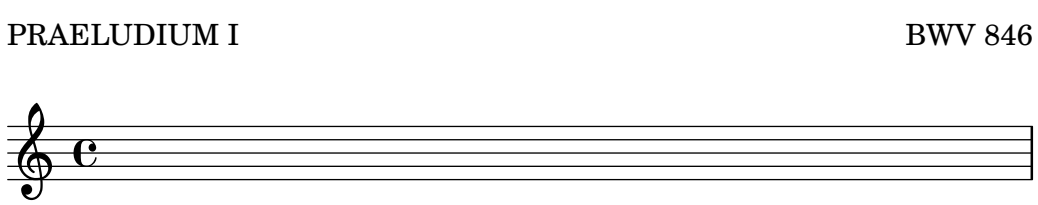
The placement of titles when using the default values of these `\markup` variables is shown in the examples in [Default layout of bookpart and score titles], pagina 459.

The default settings for `scoreTitleMarkup` as defined in '`ly/titling-init.ly`' are:

```
scoreTitleMarkup = \markup { \column {
  \on-the-fly \print-all-headers { \bookTitleMarkup \hspace #1 }
  \fill-line {
    \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
}
```

This places the `piece` and `opus` text fields at opposite ends of the same line:

```
\score {
  { s1 }
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
}
```



This example redefines `scoreTitleMarkup` so that the `piece` text field is centered and in a large, bold font.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
```

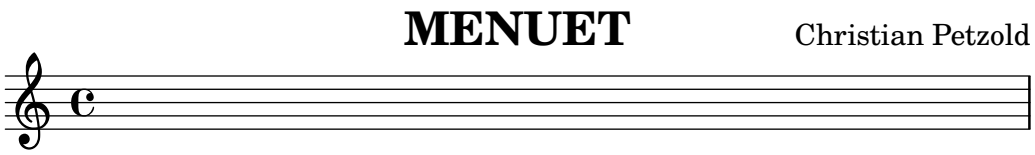
```
  \score {
    { s1 }
    \header {
      piece = "PRAELUDIUM I"
      opus = "BWV 846"
    }
  }
}
```
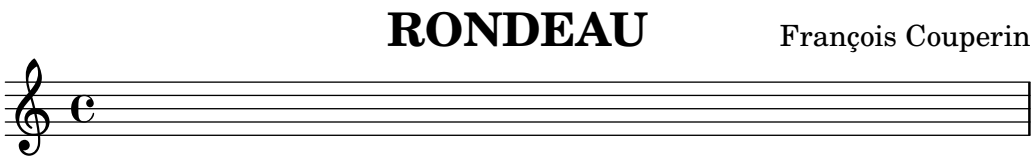


Text fields not normally effective in score `\header` blocks can be printed in the Score Title area if `print-all-headers` is placed inside the `\paper` block. A disadvantage of using this method is that text fields that are intended specifically for the Bookpart Title area need to be manually suppressed in every `\score` block. See [Titles explained], pagina 456.

To avoid this, add the desired text field to the `scoreTitleMarkup` definition. In the following example, the `composer` text field (normally associated with `bookTitleMarkup`) is added to `scoreTitleMarkup`, allowing each score to list a different composer:
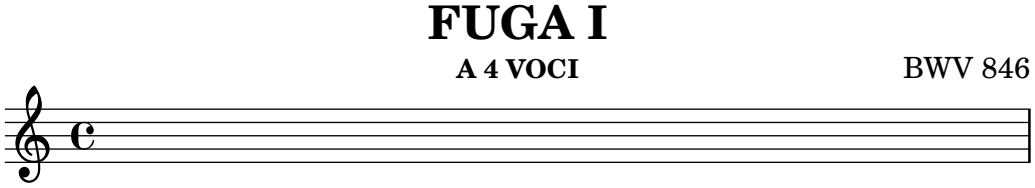
```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:composer
      }
    }
  }
  \header { tagline = ##f }
  \score {
    { s1 }
    \header {
      piece = "MENUET"
      composer = "Christian Petzold"
    }
  }
  \score {
    { s1 }
    \header {
      piece = "RONDEAU"
      composer = "François Couperin"
    }
  }
}
```





It is also possible to create your own custom text fields, and refer to them in the markup definition.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #`(direction . ,UP) {
          \dir-column {
            \center-align \fontsize #-1 \bold
              \fromproperty #'header:mycustomtext %% User-defined field
            \center-align \fontsize #4 \bold
              \fromproperty #'header:piece
          }
        }
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    { s1 }
    \header {
      piece = "FUGA I"
      mycustomtext = "A 4 VOCI" %% User-defined field
      opus = "BWV 846"
    }
  }
}
```



Vedi anche

Notation Reference: [Titles explained], pagina 456.

Custom layout for headers and footers

`\markup` commands in the `\header` block are useful for simple text formatting, but they do not allow precise control over the placement of headers and footers. To customize the placement of the text fields, use either or both of the following `\paper` variables:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

The `\markup` command `\on-the-fly` can be used to add markup conditionally to header and footer text defined within the `\paper` block, using the following syntax:

```
variable = \markup {
  ...
  \on-the-fly  \procedure  markup
  ...
}
```

The *procedure* is called each time the `\markup` command in which it appears is evaluated. The *procedure* should test for a particular condition and interpret (i.e. print) the *markup* argument if and only if the condition is true.

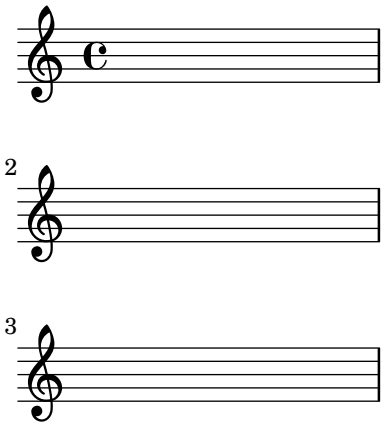
A number of ready-made procedures for testing various conditions are provided:

Procedure name	**Condition tested**
print-page-number-check-first	should this page number be printed?
create-page-number-stencil	print-page-numbers true?
print-all-headers	print-all-headers true?
first-page	first page in the book?
(on-page nmbr)	page number = nmbr?
last-page	last page in the book?
not-first-page	not first page in the book?
part-first-page	first page in the book part?
part-last-page	last page in the book part?
not-single-page	pages in book part > 1?

The following example centers page numbers at the bottom of every page. First, the default settings for `oddHeaderMarkup` and `evenHeaderMarkup` are removed by defining each as a *null* markup. Then, `oddFooterMarkup` is redefined with the page number centered. Finally, `evenFooterMarkup` is given the same layout by defining it as `\oddFooterMarkup`:

```
\book {
  \paper {
    print-page-number = ##t
    print-first-page-number = ##t
    oddHeaderMarkup = \markup \null
    evenHeaderMarkup = \markup \null
    oddFooterMarkup = \markup {
      \fill-line {
        \on-the-fly \print-page-number-check-first
        \fromproperty #'page:page-number-string
      }
```

```
    }
    evenFooterMarkup = \oddFooterMarkup
  }
  \score {
    \new Staff { s1 \break s1 \break s1 }
  }
}
```



1

Several `\on-the-fly` conditions can be combined with an 'and' operation, for example,

```
\on-the-fly \first-page
\on-the-fly \last-page
{ \markup ... \fromproperty #'header: ... }
```

determines if the output is a single page.

Vedi anche

Notation Reference: [Titles explained], pagina 456, [Default layout of bookpart and score titles], pagina 459.

Installed Files: '`../ly/titling-init.ly`'.

3.2.3 Creating footnotes

Footnotes may be used in many different situations. In all cases, a 'footnote mark' is placed as a reference in text or music, and the corresponding 'footnote text' appears at the bottom of the same page.

Footnotes within music expressions and footnotes in stand-alone text outside music expressions are created in different ways.

Footnotes in music expressions

Music footnotes overview

Footnotes in music expressions fall into two categories:

Event-based footnotes
: are attached to a particular event. Examples for such events are single notes, articulations (like fingering indications, accents, dynamics), and post-events (like slurs and manual beams). The general form for event-based footnotes is as follows:

```
[direction] \footnote [mark] offset footnote music
```

Time-based footnotes
: are bound to a particular point of time in a musical context. Some commands like `\time` and `\clef` don't actually use events for creating objects like time signatures and clefs. Neither does a chord create an event of its own: its stem or flag is created at the end of a time step (nominally through one of the note events inside). Exactly which of a chord's multiple note events will be deemed the root cause of a stem or flag is undefined. So for annotating those, time-based footnotes are preferable as well.

  A time-based footnote allows such layout objects to be annotated without referring to an event. The general form for Time-based footnotes is:

  `\footnote [mark] offset footnote [Context].GrobName`

The elements for both forms are:

direction If (and only if) the `\footnote` is being applied to a post-event or articulation, it must be preceded with a direction indicator (`-`, `_`, `^`) in order to attach *music* (with a footnote mark) to the preceding note or rest.

mark is a markup or string specifying the footnote mark which is used for marking both the reference point and the footnote itself at the bottom of the page. It may be omitted (or equivalently replaced with `\default`) in which case a number in sequence will be generated automatically. Such numerical sequences restart on each page containing a footnote.

offset is a number pair such as '`#(2 . 1)`' specifying the X and Y offsets in units of staff-spaces from the boundary of the object where the mark should be placed. Positive values of the offsets are taken from the right/top edge, negative values from the left/bottom edge and zero implies the mark is centered on the edge.

Context is the context in which the grob being footnoted is created. It may be omitted if the grob is in a bottom context, e.g. a `Voice` context.

GrobName
: specifies a type of grob to mark (like '`Flag`'). If it is specified, the footnote is not attached to a music expression in particular, but rather to all grobs of the type specified which occur at that moment of musical time.

footnote is the markup or string specifying the footnote text to use at the bottom of the page.

music is the music event or post-event or articulation that is being annotated.
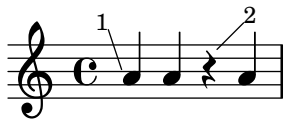
Event-based footnotes

A footnote may be attached to a layout object directly caused by the event corresponding to *music* with the syntax:

```
\footnote [mark] offset footnote music
```

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . 3) "A note" a4
    a4
    \footnote #'(2 . 2) "A rest" r4
    a4
  }
```

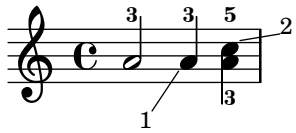
```
}
```



[1] A note
[2] A rest

Marking a *whole* chord with an event-based footnote is not possible: a chord, even one containing just a single note, does not produce an actual event of its own. However, individual notes *inside* of the chord can be marked:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(2 . 3) "Does not work" <a-3>2
    <\footnote #'(-2 . -3) "Does work" a-3>4
    <a-3 \footnote #'(3 . 1/2) "Also works" c-5>4
  }
}
```



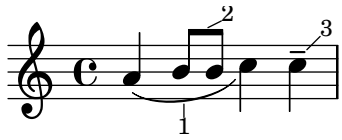
[1] Does work
[2] Also works

If the footnote is to be attached to a post-event or articulation the `\footnote` command *must* be preceded by a direction indicator, `-, _, ^`, and followed by the post-event or articulation to be annotated as the *music* argument. In this form the `\footnote` can be considered to be simply a copy of its last argument with a footnote mark attached to it. The syntax is:

direction \footnote [*mark*] *offset footnote music*

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    a4_\footnote #'(0 . -1) "A slur forced down" (
    b8^\footnote #'(1 . 0.5) "A manual beam forced up" [
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto" --
  }
```

```
}
```

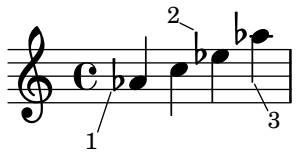


[1]A slur forced down
[2]A manual beam forced up
[3]Tenuto

Time-based footnotes

If the layout object being footmarked is *indirectly* caused by an event (like an `Accidental` or `Stem` caused by a `NoteHead` event), the *GrobName* of the layout object is required after the footnote text instead of *music*:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . -3) "A flat" Accidental
    aes4 c
    \footnote #'(-1 . 0.5) "Another flat" Accidental
    ees
    \footnote #'(1 . -2) "A stem" Stem
    aes
  }
}
```

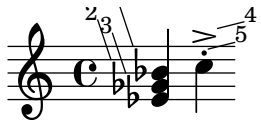


[1]A flat
[2]Another flat
[3]A stem

Note, however, that when a GrobName is specified, a footnote will be attached to all grobs of that type at the current time step:

```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote #'(-1 . 3) "A flat" Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Articulation" Script
    c'->-.
```
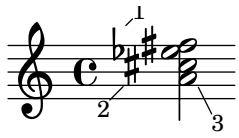
```
  }
}
```



[1]A flat
[2]A flat
[3]A flat
[4]Articulation
[5]Articulation

A note inside of a chord can be given an individual (event-based) footnote. A '`NoteHead`' is the only grob directly caused from a chord note, so an event-based footnote command is *only* suitable for adding a footnote to the '`NoteHead`' within a chord. All other chord note grobs are indirectly caused. The `\footnote` command itself offers no syntax for specifying *both* a particular grob type *as well as* a particular event to attach to. However, one can use a time-based `\footnote` command for specifying the grob type, and then prefix this command with `\single` in order to have it applied to just the following event:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    < \footnote #'(1 . -2) "An A" a
      \single \footnote #'(-1 . -1) "A sharp" Accidental
      cis
      \single \footnote #'(0.5 . 0.5) "A flat" Accidental
      ees fis
    >2
  }
}
```



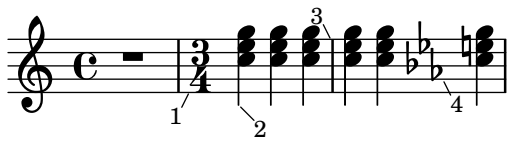
[1]A flat
[2]A sharp
[3]An A

Nota: When footnotes are attached to several musical elements at the same musical moment, as they are in the example above, the footnotes are numbered from the higher to the lower elements as they appear in the printed output, not in the order in which they are written in the input stream.

Layout objects like clefs and key-change signatures are mostly caused as a consequence of changed properties rather than actual events. Others, like bar lines and bar numbers, are a direct consequence of timing. For this reason, footnotes on such objects have to be based on their musical timing. Time-based footnotes are also preferable when marking features like stems and beams on *chords*: while such per-chord features are nominally assigned to *one* event inside the chord, relying on a particular choice would be imprudent.

The layout object in question must always be explicitly specified for time-based footnotes, and the appropriate context must be specified if the grob is created in a context other than the bottom context.

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    r1 |
    \footnote #'(-0.5 . -1) "Meter change" Staff.TimeSignature
    \time 3/4
    \footnote #'(1 . -1) "Chord stem" Stem
    <c e g>4 q q
    \footnote #'(-0.5 . 1) "Bar line" Staff.BarLine
    q q
    \footnote #'(0.5 . -1) "Key change" Staff.KeySignature
    \key c \minor
    q
  }
}
```

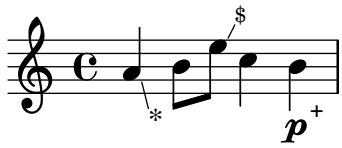


[1]Meter change
[2]Chord stem
[3]Bar line
[4]Key change

Custom marks can be used as alternatives to numerical marks, and the annotation line joining the marked object to the mark can be suppressed:

```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote "*" #'(0.5 . -2) \markup { \italic "* The first note" } a'4
    b8
    \footnote \markup { \super "$" } #'(0.5 . 1)
      \markup { \super "$" \italic " The second note" } e
    c4
    \once \override Score.FootnoteItem.annotation-line = ##f
    b-\footnote \markup \tiny "+" #'(0.1 . 0.1)
      \markup { \super "+" \italic " Editorial" } \p
  }
```

```
}
```



* *The first note*

$ *The second note*

+ *Editorial*

More examples of custom marks are shown in [Footnotes in stand-alone text], pagina 474.

Footnotes in stand-alone text

These are for use in markup outside of music expressions. They do not have a line drawn to their point of reference: their marks simply follow the referenced markup. Marks can be inserted automatically, in which case they are numerical. Alternatively, custom marks can be provided manually.

Footnotes to stand-alone text with automatic and custom marks are created in different ways.

Footnotes in stand-alone text with automatic marks

The syntax of a footnote in stand-alone text with automatic marks is

```
\markup { ... \auto-footnote text footnote ... }
```

The elements are:
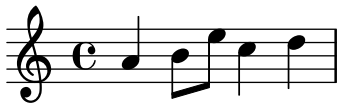
text is the markup or string to be marked.

footnote is the markup or string specifying the footnote text to use at the bottom of the page.

For example:

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple"
    \auto-footnote "tune" \italic " By me"
    "is shown below.  It is a"
    \auto-footnote "recent" \italic " Aug 2012"
    "composition."
  }
  \relative c' {
    a'4 b8 e c4 d
  }
}
```

A simple tune[1] is shown below. It is a recent[2] composition.



[1] *By me*
[2] *Aug 2012*

Footnotes in stand-alone text with custom marks

The syntax of a footnote in stand-alone text with custom marks is

```
\markup { ... \footnote mark footnote ... }
```

The elements are:
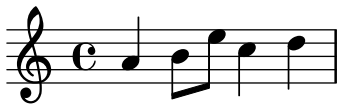
mark is a markup or string specifying the footnote mark which is used for marking the reference point. Note that this mark is *not* inserted automatically before the footnote itself.

footnote is the markup or string specifying the footnote text to use at the bottom of the page, preceded by the *mark*.

Any easy-to-type character such as * or + may be used as a mark, as shown in [Footnotes in music expressions], pagina 468. Alteratively, ASCII aliases may be used (see [ASCII aliases], pagina 488):

```
\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote "*" \italic "* By me"
    "is shown below.  It is a recent"
    \footnote \super &dagger; \concat {
      \super &dagger; \italic " Aug 2012"
    }
    "composition."
  }
  \relative c' {
    a'4 b8 e c4 d
  }
}
```

A simple tune * is shown below. It is a recent † composition.
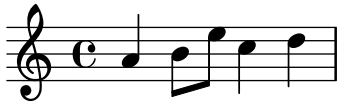


* *By me*
† *Aug 2012*

Unicode character codes may also be used to specify marks (see [Unicode], pagina 487):

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote \super \char##x00a7 \concat {
      \super \char##x00a7 \italic " By me"
    }
    "is shown below.  It is a recent"
    \footnote \super \char##x00b6 \concat {
      \super \char##x00b6 \italic " Aug 2012"
    }
    "composition."
  }
  \relative c' {
    a'4 b8 e c4 d
  }
}
```

A simple tune § is shown below. It is a recent ¶ composition.



§ *By me*
¶ *Aug 2012*

Vedi anche

Learning Manual: Sezione “Objects and interfaces” in *Manuale di Apprendimento*.

Notation Reference: [ASCII aliases], pagina 488, ⟨undefined⟩ [Balloon help], pagina ⟨undefined⟩, Sezione A.12 [List of special characters], pagina 704, ⟨undefined⟩ [Text marks], pagina ⟨undefined⟩, ⟨undefined⟩ [Text scripts], pagina ⟨undefined⟩, [Unicode], pagina 487.

Internals Reference: Sezione "FootnoteEvent" in *Guida al Funzionamento Interno*, Sezione "FootnoteItem" in *Guida al Funzionamento Interno*, Sezione "FootnoteSpanner" in *Guida al Funzionamento Interno*, Sezione "Footnote_engraver" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

Multiple footnotes for the same page can only be stacked, one above the other; they cannot be printed on the same line.

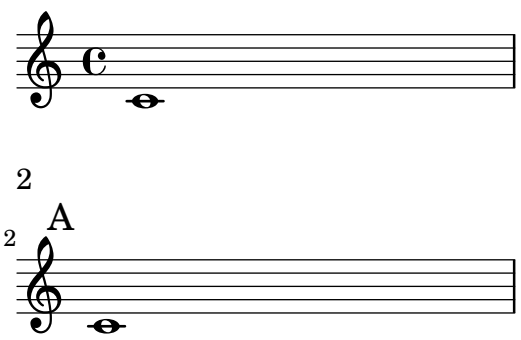
Footnotes cannot be attached to `MultiMeasureRests` or automatic beams or lyrics.

Footnote marks may collide with staves, `\markup` objects, other footnote marks and annotation lines.

3.2.4 Reference to page numbers

A particular place of a score can be marked using the `\label` command, either at top-level or inside music. This label can then be referred to in a markup, to get the number of the page where the marked point is placed, using the `\page-ref` markup command.

```
\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
    {
      c'1
      \pageBreak \mark A \label #'markA
      c'1
    }
  }
  \markup { The first score begins on page \page-ref #'firstScore "0" "?" }
  \markup { Mark A is on page \page-ref #'markA "0" "?" }
}
```



The first score begins on page 1
Mark A is on page 2

The `\page-ref` markup command takes three arguments:

1. the label, a scheme symbol, eg. `#'firstScore`;

2. a markup that will be used as a gauge to estimate the dimensions of the markup;
3. a markup that will be used in place of the page number if the label is not known;

The reason why a gauge is needed is that, at the time markups are interpreted, the page breaking has not yet occurred, so the page numbers are not yet known. To work around this issue, the actual markup interpretation is delayed to a later time; however, the dimensions of the markup have to be known before, so a gauge is used to decide these dimensions. If the book has between 10 and 99 pages, it may be "00", ie. a two digit number.

Comandi predefiniti

`\label`, `\page-ref`.

3.2.5 Table of contents

A table of contents is included using the `\markuplist \table-of-contents` command. The elements which should appear in the table of contents are entered with the `\tocItem` command, which may be used either at top-level, or inside a music expression.

```
\markuplist \table-of-contents
\pageBreak

\tocItem \markup "First score"
\score {
  {
    c'4  % ...
    \tocItem \markup "Some particular point in the first score"
    d'4  % ...
  }
}

\tocItem \markup "Second score"
\score {
  {
    e'4 % ...
  }
}
```

The markups which are used to format the table of contents are defined in the `\paper` block. The default ones are `tocTitleMarkup`, for formatting the title of the table, and `tocItemMarkup`, for formatting the toc elements, composed of the element title and page number. These variables may be changed by the user:

```
\paper {
  %% Translate the toc title into French:
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
  %% use larger font size
  tocItemMarkup = \markup \large \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
}
```

Note how the toc element text and page number are referred to in the `tocItemMarkup` definition.

New commands and markups may also be defined to build more elaborated table of contents:

- first, define a new markup variable in the `\paper` block
- then, define a music function which aims at adding a toc element using this markup paper variable.

In the following example, a new style is defined for entering act names in the table of contents of an opera:

```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}

tocAct =
#(define-music-function (parser location text) (markup?)
   (add-toc-item! 'tocActMarkup text))
```

Table of Contents

Atto Primo

Coro. Viva il nostro Alcide 1
Cesare. Presti omai l'Egizzia terra 1

Atto Secondo

Sinfonia 1
Cleopatra. V'adoro, pupille, saette d'Amore 1

Dots can be added to fill the line between an item and its page number:

```
\header { tagline = ##f }
\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}

\book {
  \markuplist \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}
```

Table of Contents

Allegro 1
Largo 1

Vedi anche

Installed Files: ‘`ly/toc-init.ly`’.

Comandi predefiniti

`\table-of-contents`, `\tocItem`.

3.3 Working with input files

3.3.1 Including LilyPond files

A large project may be split up into separate files. To refer to another file, use

```
\include "otherfile.ly"
```

The line `\include "otherfile.ly"` is equivalent to pasting the contents of ‘`otherfile.ly`’ into the current file at the place where the `\include` appears. For example, in a large project you might write separate files for each instrument part and create a “full score” file which brings together the individual instrument files. Normally the included file will define a number of variables which then become available for use in the full score file. Tagged sections can be marked in included files to assist in making them usable in different places in a score, see Sezione 3.3.2 [Different editions from one source], pagina 481.

Files in the current working directory may be referenced by specifying just the file name after the `\include` command. Files in other locations may be included by giving either a full path reference or a relative path reference (but use the UNIX forward slash, /, rather than the DOS/Windows back slash, \, as the directory separator.) For example, if ‘`stuff.ly`’ is located one directory higher than the current working directory, use

```
\include "../stuff.ly"
```

or if the included orchestral parts files are all located in a subdirectory called ‘`parts`’ within the current directory, use

```
\include "parts/VI.ly"
\include "parts/VII.ly"
... etc
```

Files which are to be included can also contain `\include` statements of their own. By default, these second-level `\include` statements are not interpreted until they have been brought into the main file, so the file names they specify must all be relative to the directory containing the main file, not the directory containing the included file. However, this behavior can be changed globally by passing the option ‘`-drelative-includes`’ option at the command line (or by adding `#(ly:set-option 'relative-includes #t)` at the top of the main input file).

When `relative-includes` is set to `#t`, the path for each `\include` command will be taken relative to the file containing that command. This behavior is recommended and it will become the default behavior in a future version of lilypond.

Files relative to the main directory and files relative to some other directory may both be `\included` by setting `relative-includes` to `#t` or `#f` at appropriate places in the files. For example, if a general library, libA, has been created which itself uses sub-files which are `\included` by the entry file of that library, those `\include` statements will need to be preceded by `#(ly:set-option #relative-includes #t)` so they are interpreted correctly when brought into the main `.ly` file, like this:

```
libA/
  libA.ly
  A1.ly
  A2.ly
  ...
```

then the entry file, `libA.ly`, will contain

```
#(ly:set-option 'relative-includes #t)
\include "A1.ly"
\include "A2.ly"
...
% return to default setting
#(ly:set-option 'relative-includes #f)
```

Any '`.ly`' file can then include the entire library simply with

```
\include "~/libA/libA.ly"
```

More complex file structures may be devised by switching at appropriate places.

Files can also be included from a directory in a search path specified as an option when invoking LilyPond from the command line. The included files are then specified using just their file name. For example, to compile '`main.ly`' which includes files located in a subdirectory called '`parts`' by this method, cd to the directory containing '`main.ly`' and enter

```
lilypond --include=parts main.ly
```

and in main.ly write

```
\include "VI.ly"
\include "VII.ly"
... etc
```

Files which are to be included in many scores may be placed in the LilyPond directory '`../ly`'. (The location of this directory is installation-dependent - see Sezione "Other sources of information" in *Manuale di Apprendimento*). These files can then be included simply by naming them on an `\include` statement. This is how the language-dependent files like '`english.ly`' are included.

LilyPond includes a number of files by default when you start the program. These includes are not apparent to the user, but the files may be identified by running `lilypond --verbose` from the command line. This will display a list of paths and files that LilyPond uses, along with much other information. Alternatively, the more important of these files are discussed in Sezione "Other sources of information" in *Manuale di Apprendimento*. These files may be edited, but changes to them will be lost on installing a new version of LilyPond.

Some simple examples of using `\include` are shown in Sezione "Scores and parts" in *Manuale di Apprendimento*.

Vedi anche

Learning Manual: Sezione "Other sources of information" in *Manuale di Apprendimento*, Sezione "Scores and parts" in *Manuale di Apprendimento*.

Problemi noti e avvertimenti

If an included file is given a name which is the same as one in LilyPond's installation files, LilyPond's file from the installation files takes precedence.

3.3.2 Different editions from one source
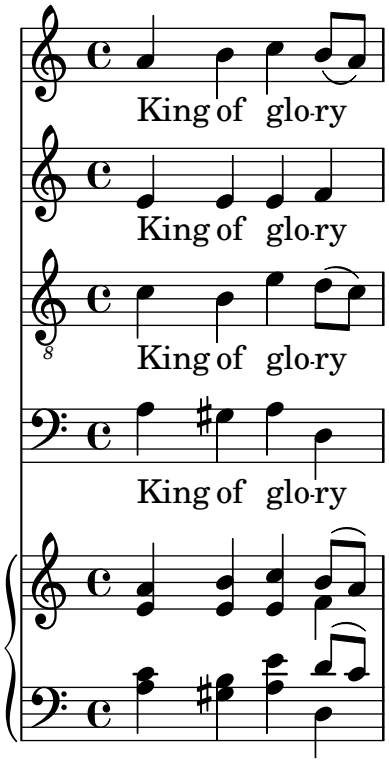
Several methods can be used to generate different versions of a score from the same music source. Variables are perhaps the most useful for combining lengthy sections of music and/or annotation. Tags are more useful for selecting one section from several alternative shorter sections of music, and can also be used for splicing pieces of music together at different points.

Whichever method is used, separating the notation from the structure of the score will make it easier to change the structure while leaving the notation untouched.

Using variables

If sections of the music are defined in variables they can be reused in different parts of the score, see Sezione “Organizing pieces with variables” in *Manuale di Apprendimento*. For example, an *a cappella* vocal score frequently includes a piano reduction of the parts for rehearsal purposes which is identical to the vocal music, so the music need be entered only once. Music from two variables may be combined on one staff, see ⟨undefined⟩ [Automatic part combining], pagina ⟨undefined⟩. Here is an example:

```
sopranoMusic = \relative c'' { a4 b c b8( a) }
altoMusic = \relative g' { e4 e e f }
tenorMusic = \relative c' { c4 b e d8( c) }
bassMusic = \relative c' { a4 gis a d, }
allLyrics = \lyricmode {King of glo -- ry }
<<
  \new Staff = "Soprano" \sopranoMusic
  \new Lyrics \allLyrics
  \new Staff = "Alto" \altoMusic
  \new Lyrics \allLyrics
  \new Staff = "Tenor" {
    \clef "treble_8"
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Bass" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "RH" {
      \set Staff.printPartCombineTexts = ##f
      \partcombine
      \sopranoMusic
      \altoMusic
    }
    \new Staff = "LH" {
      \set Staff.printPartCombineTexts = ##f
      \clef "bass"
      \partcombine
      \tenorMusic
      \bassMusic
    }
  >>
>>
```



Separate scores showing just the vocal parts or just the piano part can be produced by changing just the structural statements, leaving the musical notation unchanged.

For lengthy scores, the variable definitions may be placed in separate files which are then included, see Sezione 3.3.1 [Including LilyPond files], pagina 480.

Using tags

The `\tag #'partA` command marks a music expression with the name *partA*. Expressions tagged in this way can be selected or filtered out by name later, using either `\keepWithTag #'name` or `\removeWithTag #'name`. The result of applying these filters to tagged music is as follows:

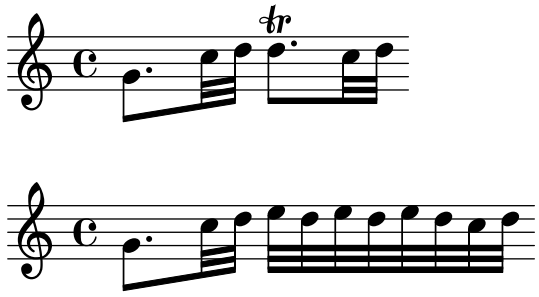
Filter	**Result**
Tagged music preceded by `\keepWithTag #'name` or `\keepWithTag #'(name1 name2...)`	Untagged music and music tagged with any of the given tag names is included; music tagged with any other tag name is excluded.
Tagged music preceded by `\removeWithTag #'name` or `\removeWithTag #'(name1 name2...)`	Untagged music and music not tagged with any of the given tag names is included; music tagged with any of the given tag names is excluded.
Tagged music not preceded by either `\keepWithTag` or `\removeWithTag`	All tagged and untagged music is included.

The arguments of the `\tag`, `\keepWithTag` and `\removeWithTag` commands should be a symbol (such as `#'score` or `#'part`), followed by a music expression.

In the following example, we see two versions of a piece of music, one showing trills with the usual notation, and one with trills explicitly expanded:

```
music = \relative g' {
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand { \repeat unfold 3 { e32 d } }
  c32 d
 }
```

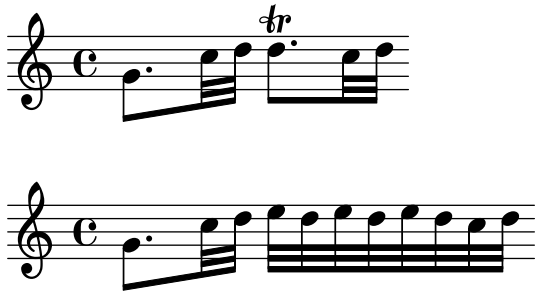
```
\score {
  \keepWithTag #'trills \music
}
\score {
  \keepWithTag #'expand \music
}
```



Alternatively, it is sometimes easier to exclude sections of music:

```
music = \relative g' {
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand {\repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \removeWithTag #'expand
  \music
}
\score {
  \removeWithTag #'trills
  \music
}
```



Tagged filtering can be applied to articulations, texts, etc. by prepending

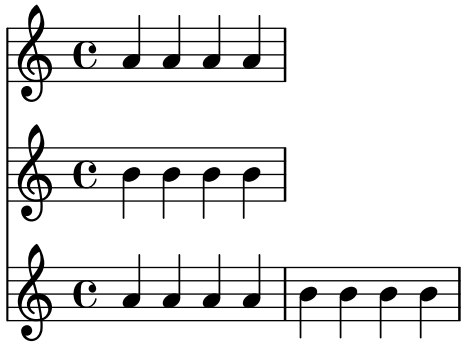
`-\tag #'`*your-tag*

to an articulation. For example, this would define a note with a conditional fingering indication and a note with a conditional annotation:

```
c1-\tag #'finger ^4
c1-\tag #'warn ^"Watch!"
```
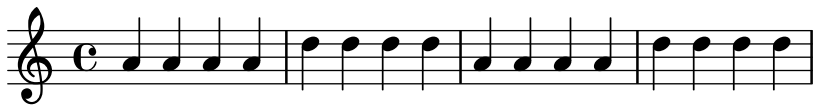
Multiple tags may be placed on expressions with multiple `\tag` entries, or by combining multiple tags into one symbol list:

```
music = \relative c'' {
  \tag #'a \tag #'both { a4 a a a }
  \tag #'(b both) { b4 b b b }
}
<<
\keepWithTag #'a \music
\keepWithTag #'b \music
\keepWithTag #'both \music
>>
```



Multiple `\removeWithTag` filters may be applied to a single music expression to remove several differently named tagged sections. Alternatively, you can use a single `\removeWithTag` with a list of tags.

```
music = \relative c'' {
  \tag #'A { a4 a a a }
  \tag #'B { b4 b b b }
  \tag #'C { c4 c c c }
  \tag #'D { d4 d d d }
}
\new Voice {
  \removeWithTag #'B
  \removeWithTag #'C
  \music
  \removeWithTag #'(B C)
  \music
}
```
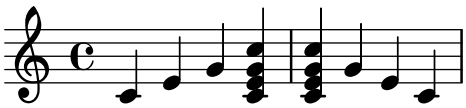


Two or more `\keepWithTag` filters applied to a single music expression will cause *all* tagged sections to be removed, as the first filter will remove all tagged sections except the one named, and the second filter will remove even that tagged section. Usually you would rather want to use a single `\keepWithTag` command with a list of multiple tags: this will only remove tagged sections not given in *any* of the tags.

Sometimes you want to splice some music at a particular place in an existing music expression. You can use `\pushToTag` and `\appendToTag` for adding material at the front or end of the `elements` of an existing music construct. Not every music construct has `elements`, but sequential and simultaneous music are safe bets:

```
test = { \tag #'here { \tag #'here <<c''>> } }

{
  \pushToTag #'here c'
  \pushToTag #'here e'
  \pushToTag #'here g' \test
  \appendToTag #'here c'
  \appendToTag #'here e'
  \appendToTag #'here g' \test
}
```



Both commands get a tag, the material to splice in at every occurence of the tag, and the tagged expression. The commands make sure to copy everything that they change so that the original `\test` retains its meaning.

Vedi anche

Learning Manual: Sezione “Organizing pieces with variables” in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Automatic part combining], pagina ⟨undefined⟩, Sezione 3.3.1 [Including LilyPond files], pagina 480.

Problemi noti e avvertimenti

Calling `\relative` on a music expression obtained by filtering music through `\keepWithTag` or `\removeWithTag` might cause the octave relations to change, as only the pitches actually remaining in the filtered expression will be considered. Applying `\relative` first, before `\keepWithTag` or `\removeWithTag`, avoids this danger as `\relative` then acts on all the pitches as-input.

Using global settings

Global settings can be included from a separate file:

```
lilypond -dinclude-settings=MY_SETTINGS.ly MY_SCORE.ly
```

Groups of settings such as page size, font or type face can be stored in separate files. This allows different editions from the same score as well as standard settings to be applied to many scores, simply by specifying the proper settings file.

This technique also works well with the use of style sheets, as discussed in Sezione “Style sheets” in *Manuale di Apprendimento*.

Vedi anche

Learning Manual: Sezione “Organizing pieces with variables” in *Manuale di Apprendimento*, Sezione “Style sheets” in *Manuale di Apprendimento*.

Notation Reference: Sezione 3.3.1 [Including LilyPond files], pagina 480.

3.3.3 Special characters

Text encoding

LilyPond uses the character repertoire defined by the Unicode consortium and ISO/IEC 10646. This defines a unique name and code point for the character sets used in virtually all modern languages and many others too. Unicode can be implemented using several different encodings.

LilyPond uses the UTF-8 encoding (UTF stands for Unicode Transformation Format) which represents all common Latin characters in one byte, and represents other characters using a variable length format of up to four bytes.

The actual appearance of the characters is determined by the glyphs defined in the particular fonts available - a font defines the mapping of a subset of the Unicode code points to glyphs. LilyPond uses the Pango library to layout and render multi-lingual texts.
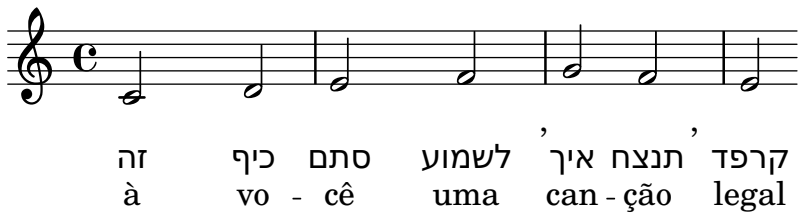
LilyPond does not perform any input-encoding conversions. This means that any text, be it title, lyric text, or musical instruction containing non-ASCII characters, must be encoded in UTF-8. The easiest way to enter such text is by using a Unicode-aware editor and saving the file with UTF-8 encoding. Most popular modern editors have UTF-8 support, for example, vim, Emacs, jEdit, and GEdit do. All MS Windows systems later than NT use Unicode as their native character encoding, so even Notepad can edit and save a file in UTF-8 format. A more functional alternative for Windows is BabelPad.

If a LilyPond input file containing a non-ASCII character is not saved in UTF-8 format the error message

```
FT_Get_Glyph_Name () error: invalid argument
```

will be generated.

Here is an example showing Cyrillic, Hebrew and Portuguese text:



Unicode

To enter a single character for which the Unicode code point is known but which is not available in the editor being used, use either `\char ##xhhhh` or `\char #dddd` within a `\markup` block, where `hhhh` is the hexadecimal code for the character required and `dddd` is the corresponding decimal value. Leading zeroes may be omitted, but it is usual to specify all four characters in the hexadecimal representation. (Note that the UTF-8 encoding of the code point should *not* be used after `\char`, as UTF-8 encodings contain extra bits indicating the number of octets.) Unicode code charts and a character name index giving the code point in hexadecimal for any character can be found on the Unicode Consortium website, http://www.unicode.org/.

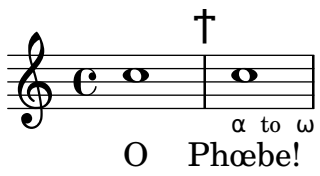
For example, `\char ##x03BE` and `\char #958` would both enter the Unicode U+03BE character, which has the Unicode name "Greek Small Letter Xi".

Any Unicode code point may be entered in this way and if all special characters are entered in this format it is not necessary to save the input file in UTF-8 format. Of course, a font containing all such encoded characters must be installed and available to LilyPond.

The following example shows Unicode hexadecimal values being entered in four places – in a rehearsal mark, as articulation text, in lyrics and as stand-alone text below the score:

```
\score {
  \relative c'' {
    c1 \mark \markup { \char ##x03EE }
    c1_\markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } }
  }
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } }
}
```

```
\markup { "Copyright 2008--2012" \char ##x00A9 }
```



Copyright 2008--2012 ©

To enter the copyright sign in the copyright notice use:

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

ASCII aliases

A list of ASCII aliases for special characters can be included:

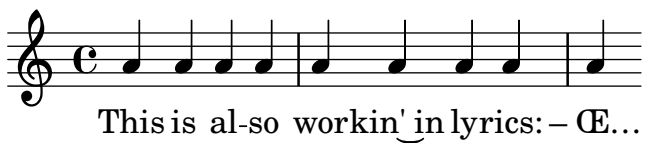
```
\paper {
  #(include-special-characters)
}

\markup "&flqq; &ndash; &OE;uvre incomplète&hellip; &frqq;"

\score {
  \new Staff { \repeat unfold 9 a'4 }
  \addlyrics {
    This is al -- so wor -- kin'~in ly -- rics: &ndash;_&OE;&hellip;
  }
}

\markup \column {
  "The replacement can be disabled:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}
```

« – Œuvre incomplète… »



The replacement can be disabled:
– Œ …
– &OE; …

You can also make your own aliases, either globally:

```
\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}
\markup "A 100 dpi."
```

A hundred dots per inch.

or locally:

```
\markup \replace #'(("100" . "hundred")
                    ("dpi" . "dots per inch")) "A 100 dpi."
```

A hundred dots per inch.

Vedi anche

Notation Reference: Sezione A.12 [List of special characters], pagina 704.

Installed Files: ‘`ly/text-replacements.ly`’.

3.4 Controlling output

3.4.1 Extracting fragments of music

It is possible to quote small fragments of a large score directly from the output. This can be compared to clipping a piece of a paper score with scissors.

This is done by defining the measures that need to be cut out separately. For example, including the following definition

```
\layout {
  clip-regions
  = #(list
      (cons
       (make-rhythmic-location 5 1 2)
       (make-rhythmic-location 7 3 4)))
}
```

will extract a fragment starting halfway the fifth measure, ending in the seventh measure. The meaning of `5 1 2` is: after a 1/2 note in measure 5, and `7 3 4` after 3 quarter notes in measure 7.

More clip regions can be defined by adding more pairs of rhythmic-locations to the list.

In order to use this feature, LilyPond must be invoked with ‘`-dclip-systems`’. The clips are output as EPS files, and are converted to PDF and PNG if these formats are switched on as well.

For more information on output formats, see Sezione “Invoking lilypond” in *Uso del Programma*.

3.4.2 Skipping corrected music

When entering or copying music, usually only the music near the end (where you are adding notes) is interesting to view and correct. To speed up this correction process, it is possible to skip typesetting of all but the last few measures. This is achieved by putting

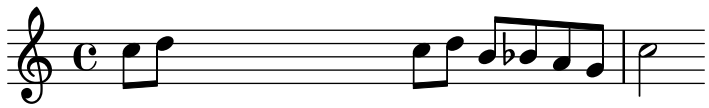
```
showLastLength = R1*5
\score { ... }
```

in your source file. This will render only the last 5 measures (assuming 4/4 time signature) of every `\score` in the input file. For longer pieces, rendering only a small part is often an order of magnitude quicker than rendering it completely. When working on the beginning of a score you have already typeset (e.g. to add a new part), the `showFirstLength` property may be useful as well.

Skipping parts of a score can be controlled in a more fine-grained fashion with the property `Score.skipTypesetting`. When it is set, no typesetting is performed at all.

This property is also used to control output to the MIDI file. Note that it skips all events, including tempo and instrument changes. You have been warned.

```
c8 d
\set Score.skipTypesetting = ##t
e8 e e e e e e e
\set Score.skipTypesetting = ##f
c8 d b bes a g c2
```



In polyphonic music, `Score.skipTypesetting` will affect all voices and staves, saving even more time.

3.4.3 Alternative output formats

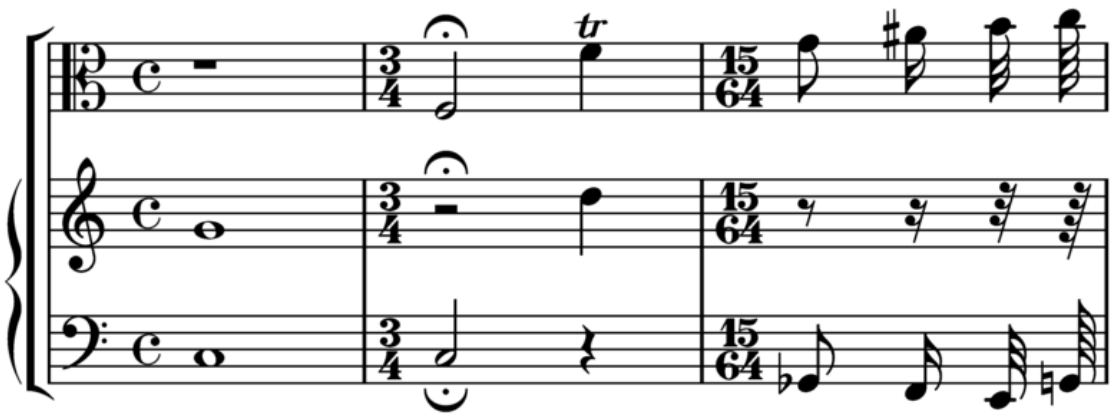
The default output formats for the printed score are Portable Document Format (PDF) and PostScript (PS). Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS) and Portable Network Graphics (PNG) output formats are also available through command line options, see Sezione “Basic command line options for LilyPond” in *Uso del Programma*.

3.4.4 Replacing the notation font
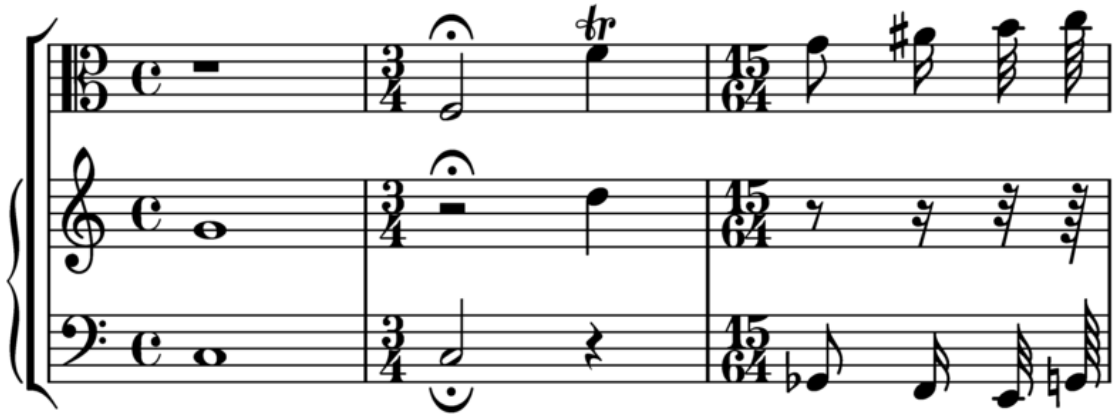
Gonville is an alternative to the Feta font used in LilyPond and can be downloaded from:

http://www.chiark.greenend.org.uk/~sgtatham/gonville/

Here are a few sample bars of music set in Gonville:



Here are a few sample bars of music set in LilyPond's Feta font:



Installation Instructions for MacOS

Download and extract the zip file. Copy the `lilyfonts` directory to '*SHARE_DIR*`/lilypond/current`'; for more information, see Sezione "Other sources of information" in *Manuale di Apprendimento*. Rename the existing `fonts` directory to `fonts_orig` and the `lilyfonts` directory to `fonts`. To revert back to Feta, reverse the process.

Vedi anche

Learning Manual: Sezione "Other sources of information" in *Manuale di Apprendimento*.

Problemi noti e avvertimenti

Gonville cannot be used to typeset 'Ancient Music' notation and it is likely newer glyphs in later releases of LilyPond may not exist in the Gonville font family. Please refer to the author's website for more information on these and other specifics, including licensing of Gonville.

3.5 MIDI output

MIDI (Musical Instrument Digital Interface) is a standard for connecting and controlling digital instruments. A MIDI file is a series of notes in a number of tracks. It is not an actual sound file; you need special software to translate between the series of notes and actual sounds.

Pieces of music can be converted to MIDI files, so you can listen to what was entered. This is convenient for checking the music; octaves that are off or accidentals that were mistyped stand out very much when listening to the MIDI output.

Standard MIDI output is somewhat crude; optionally, an enhanced and more realistic MIDI output is available by means of Sezione 3.5.7 [The Articulate script], pagina 500.

The MIDI output allocates a channel for each staff, and reserves channel 10 for drums. There are only 16 MIDI channels per device, so if the score contains more than 15 staves, MIDI channels will be reused.

3.5.1 Creating MIDI files

To create a MIDI output file from a LilyPond file, insert a `\midi` block inside a `\score` block;

```
\score {
  ...music...
  \layout { }
  \midi { }
}
```

If there is *only* a `\midi` block in a `\score` (i.e. without any `\layout` block), then *only* MIDI output will be produced. No musical notation will be typeset.

```
\score {
  ...music...
  \midi { }
}
```

Dynamics, pitches, rhythms, tempo changes and ties are all interpreted and translated correctly. Dynamic 'marks' translate into volume levels with a 'fixed fraction' of the available MIDI volume range; crescendi and decrescendi make the volume vary linearly between their two extremes.

All `\tempo` indications, including all subsequent tempo changes, specified within the music notation will be reflected in the MIDI output.

Usually it is enough to leave the `\midi` block empty, but it can contain context rearrangements, new context definitions or code to set the values of properties. Here the tempo is set to 72 quarter-note beats per minute, but *only* for the MIDI's audio playback.

```
\score {
  ...music...
  \midi {
    \tempo 4 = 72
  }
}
```

Note that `\tempo` is actually a command for setting properties during the interpretation of the music and in the context of output definitions, like a `\midi` block, is reinterpreted as if it were a context modification.

Context definitions follow the same syntax as those in a `\layout` block;

```
\score {
  ...music...
  \midi {
    \context {
      \Voice
      \remove "Dynamic_performer"
    }
  }
}
```

removes the effect of dynamics from the MIDI output. Translation modules for sound are called 'performers'.

Frammenti di codice selezionati
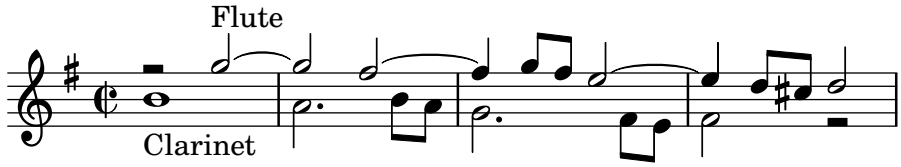
Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per MIDI port, and most devices support only one port.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `midiInstrument`.

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
```

```
      \set midiInstrument = #"flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = #"clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
    }
    \tempo 2 = 72
  }
}
```



Problemi noti e avvertimenti

Some operating systems require a *specific* file extension for MIDI files. If a different extension is preferred insert the following line at the top-level of the input file, before the start of any `\book`, `\bookpart` or `\score` blocks;

```
#(ly:set-option 'midi-extension "mid")
```

This will set the default extension for MIDI files to `.mid`.

Alternatively, an option can be supplied on the command line:

```
lilypond -dmidi-extension=mid MyFile.ly
```

Changes in the MIDI volume take place only on starting a note, so crescendi and decrescendi cannot affect the volume of a single note.

Some MIDI players may not always correctly handle tempo changes in the midi output.

Vedi anche

Installed Files: '`../ly/performer-init.ly`'.

Learning Manual: Sezione "Other sources of information" in *Manuale di Apprendimento*.

3.5.2 MIDI Instruments

The MIDI instrument to be used is specified by setting the `Staff.midiInstrument` property to the instrument name. The name should be chosen from the list in Sezione A.6 [MIDI instruments], pagina 630.

```
\new Staff {
  \set Staff.midiInstrument = #"glockenspiel"
  ...notes...
}
\new Staff \with {midiInstrument = #"cello"} {
  ...notes...
}
```

If the selected instrument does not exactly match an instrument from the list of MIDI instruments, the Grand Piano (`"acoustic grand"`) instrument is used.

3.5.3 What goes into the MIDI output?

Supported in MIDI

The following items of notation are reflected in the MIDI output:

- Pitches
- Microtones (See ⟨undefined⟩ [Accidentals], pagina ⟨undefined⟩. Rendering needs a player that supports pitch bend.)
- Chords entered as chord names
- Rhythms entered as note durations, including tuplets
- Tremolos entered without '`:`[*number*]'
- Ties
- Dynamic marks
- Crescendi, decrescendi over multiple notes
- Tempo changes entered with a tempo marking
- Lyrics

Using Sezione 3.5.7 [The Articulate script], pagina 500, a number of items are added to the above list:

- Articulations (slurs, staccato, etc)
- Trills, turns
- Rallentando and accelerando

Unsupported in MIDI

The following items of notation have no effect on the MIDI output, unless you use Sezione 3.5.7 [The Articulate script], pagina 500:

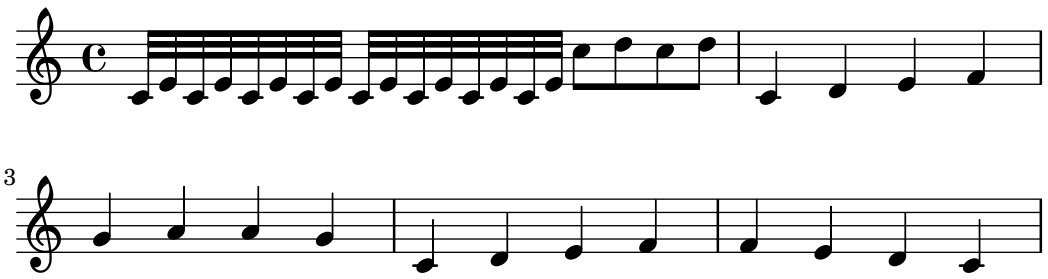
- Rhythms entered as annotations, e.g. swing
- Tempo changes entered as annotations with no tempo marking
- Staccato and other articulations and ornamentations
- Slurs and Phrasing slurs

- Crescendi, decrescendi over a single note
- Tremolos entered with ':[*number*]'
- Figured bass
- Microtonal chords

3.5.4 Repeats in MIDI

With a few minor additions, all types of repeats can be represented in the MIDI output. This is achieved by applying the `\unfoldRepeats` music function. This function changes all repeats to unfold repeats.

```
\unfoldRepeats {
  \repeat tremolo 8 { c'32 e' }
  \repeat percent 2 { c''8 d'' }
  \repeat volta 2 { c'4 d' e' f' }
  \alternative {
    { g' a' a' g' }
    { f' e' d' c' }
  }
}
\bar "|."
```



In scores containing multiple voices, unfolding of repeats in MIDI output will only occur correctly if *each* voice contains fully notated repeat indications.

When creating a score file using `\unfoldRepeats` for MIDI, it is necessary to make two `\score` blocks: one for MIDI (with unfolded repeats) and one for notation (with volta, tremolo, and percent repeats). For example,

```
\score {
  ...music...
  \layout { ... }
}
\score {
  \unfoldRepeats ...music...
  \midi { ... }
}
```

3.5.5 Controlling MIDI dynamics

MIDI dynamics are implemented by the Dynamic_performer which lives by default in the Voice context. It is possible to control the overall MIDI volume, the relative volume of dynamic markings and the relative volume of different instruments.
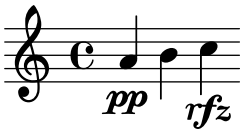
Dynamic marks

Dynamic marks are translated to a fixed fraction of the available MIDI volume range. The default fractions range from 0.25 for *ppppp* to 0.95 for *fffff*. The set of dynamic marks and the associated fractions can be seen in '`../scm/midi.scm`', see Sezione "Other sources of information" in *Manuale di Apprendimento*. This set of fractions may be changed or extended by providing a function which takes a dynamic mark as its argument and returns the required fraction, and setting `Score.dynamicAbsoluteVolumeFunction` to this function.

For example, if a *rinforzando* dynamic marking, `\rfz`, is required, this will not by default have any effect on the MIDI volume, as this dynamic marking is not included in the default set. Similarly, if a new dynamic marking has been defined with `make-dynamic-script` that too will not be included in the default set. The following example shows how the MIDI volume for such dynamic markings might be added. The Scheme function sets the fraction to 0.9 if a dynamic mark of rfz is found, or calls the default function otherwise.

```
#(define (myDynamics dynamic)
    (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative c'' {
        a4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



Alternatively, if the whole table of fractions needs to be redefined, it would be better to use the *default-dynamic-absolute-volume* procedure in '`../scm/midi.scm`' and the associated table as a model. The final example in this section shows how this might be done.

Overall MIDI volume
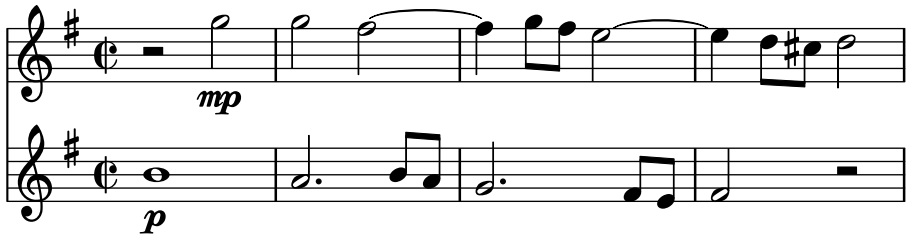
The minimum and maximum overall volume of MIDI dynamic markings is controlled by setting the properties `midiMinimumVolume` and `midiMaximumVolume` at the `Score` level. These properties have an effect only at the start of a voice and on dynamic marks. The fraction corresponding to each dynamic mark is modified with this formula

```
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) * fraction
```

In the following example the dynamic range of the overall MIDI volume is limited to the range 0.2 - 0.5.

```
\score {
  <<
```

```
    \new Staff {
      \key g \major
      \time 2/2
      \set Staff.midiInstrument = #"flute"
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout {}
  \midi {
    \tempo 2 = 72
    \context {
      \Score
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}
```



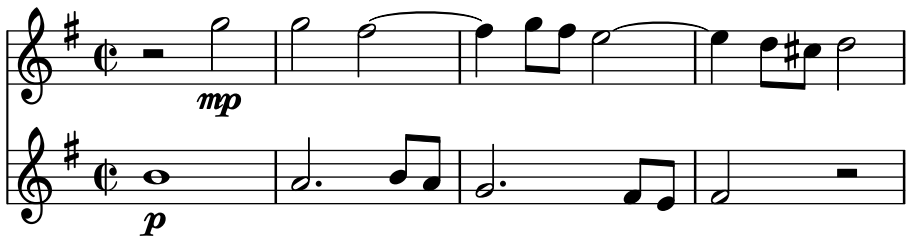
Equalizing different instruments (i)

If the minimum and maximum MIDI volume properties are set in the `Staff` context the relative volumes of the MIDI instruments can be controlled. This gives a basic instrument equalizer, which can enhance the quality of the MIDI output remarkably.

In this example the volume of the clarinet is reduced relative to the volume of the flute.

```
\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Staff.midiInstrument = #"flute"
      \set Staff.midiMinimumVolume = #0.7
```

```
      \set Staff.midiMaximumVolume = #0.9
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \set Staff.midiMinimumVolume = #0.3
      \set Staff.midiMaximumVolume = #0.6
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout {}
  \midi {
    \tempo 2 = 72
  }
}
```



Equalizing different instruments (ii)

If the MIDI minimum and maximum volume properties are not set LilyPond will, by default, apply a small degree of equalization to a few instruments. The instruments and the equalization applied are shown in the table *instrument-equalizer-alist* in '`../scm/midi.scm`'.

This basic default equalizer can be replaced by setting `instrumentEqualizer` in the `Score` context to a new Scheme procedure which accepts a MIDI instrument name as its only argument and returns a pair of fractions giving the minimum and maximum volumes to be applied to that instrument. This replacement is done in the same way as shown for resetting the `dynamicAbsoluteVolumeFunction` at the start of this section. The default equalizer, *default-instrument-equalizer*, in '`../scm/midi.scm`' shows how such a procedure might be written.

The following example sets the relative flute and clarinet volumes to the same values as the previous example.
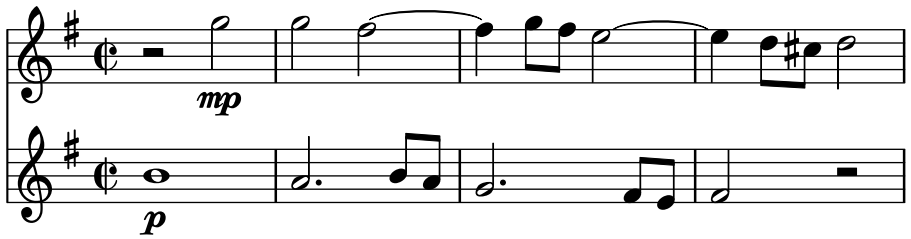
```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(
      ("flute" . (0.7 . 0.9))
```

```
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = #"flute"
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi {
    \tempo 2 = 72
  }
}
```



3.5.6 Percussion in MIDI

Percussion instruments are generally notated in a `DrumStaff` context and when notated in this way they are outputted correctly to MIDI channel 10, but some pitched percussion instruments, like the xylophone, marimba, vibraphone, timpani, etc., are treated like "normal" instruments and music for these instruments should be entered in a normal `Staff` context, not a `DrumStaff` context, to obtain the correct MIDI output.

Some non-pitched percussion sounds included in the general MIDI standard, like melodic tom, taiko drum, synth drum, etc., cannot be reached via MIDI channel 10, so the notation for such instruments should also be entered in a normal `Staff` context, using suitable normal pitches.

Many percussion instruments are not included in the general MIDI standard, e.g. castanets. The easiest, although unsatisfactory, method of producing some MIDI output when writing for such instruments is to substitute the nearest sound from the standard set.

Problemi noti e avvertimenti

Because the general MIDI standard does not contain rim shots, the sidestick is used for this purpose instead.

3.5.7 The Articulate script

A more realistic MIDI output is possible when using the Articulate script. It tries to take articulations (slurs, staccato, etc) into account, by replacing notes with sequential music of suitably time-scaled note plus skip. It also tries to unfold trills turns etc., and take rallentando and accelerando into account.

To use the Articulate script, you have to include it at the top of your input file,

```
\include "articulate.ly"
```

and in the `\score` section do

```
\unfoldRepeats \articulate <<
all the rest of the score...
>>
```

After altering your input file this way, the visual output is heavily altered, but the standard `\midi` block will produce a better MIDI file.

Although not essential for the Articulate script to work, you may want to insert the `\unfoldRepeats` command as it appears in the example shown above as it enables performing abbreviatures such as *trills*.

Problemi noti e avvertimenti

Articulate shortens chords and some music (esp. organ music) could sound worse.

3.6 Extracting musical information

In addition to creating graphical output and MIDI, LilyPond can display musical information as text.

3.6.1 Displaying LilyPond notation

Displaying a music expression in LilyPond notation can be done with the music function `\displayLilyMusic`. To see the output, you will typically want to call LilyPond using the command line. For example,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

will display

```
{ a,4 cis e fis g }
```

By default, LilyPond will print these messages to the console along with all the other LilyPond compilation messages. To split up these messages and save the results of `\displayLilyMusic`, redirect the output to a file.

```
lilypond file.ly >display.txt
```

Note that Lilypond does not just display the music expression, but also interprets it (since `\displayLilyMusic` returns it in addition to displaying it). This is convenient since you can just insert `\displayLilyMusic` into existing music in order to get information about it. If you don't actually want Lilypond to interpret the displayed music as well as display it, use `\void` in order to have it ignored:

```
{
  \void \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

3.6.2 Displaying scheme music expressions

See Sezione "Displaying music expressions" in *Estendere*.

3.6.3 Saving music events to a file

Music events can be saved to a file on a per-staff basis by including a file in your main score.

```
\include "event-listener.ly"
```

This will create file(s) called '`FILENAME-STAFFNAME.notes`' or '`FILENAME-unnamed-staff.notes`' for each staff. Note that if you have multiple unnamed staves, the events for all staves will be mixed together in the same file. The output looks like this:

```
0.000   note     57       4   p-c 2 12
0.000   dynamic  f
0.250   note     62       4   p-c 7 12
0.500   note     66       8   p-c 9 12
0.625   note     69       8   p-c 14 12
0.750   rest     4
0.750   breathe
```

The syntax is a tab-delimited line, with two fixed fields on each line followed by optional parameters.

`time  type  ...params...`

This information can easily be read into other programs such as python scripts, and can be very useful for researchers wishing to perform musical analysis or playback experiments with LilyPond.

Problemi noti e avvertimenti

Not all lilypond music events are supported by '`event-listener.ly`'. It is intended to be a well-crafted "proof of concept". If some events that you want to see are not included, copy '`event-listener.ly`' into your lilypond directory and modify the file so that it outputs the information you want.

4 Spacing issues

The global paper layout is determined by three factors: the page layout, the line breaks, and the spacing. These all influence each other. The choice of spacing determines how densely each system of music is set. This influences where line breaks are chosen, and thus ultimately, how many pages a piece of music takes.

Globally speaking, this procedure happens in four steps: first, flexible distances ('springs') are chosen, based on durations. All possible line breaking combinations are tried, and a 'badness' score is calculated for each. Then the height of each possible system is estimated. Finally, a page breaking and line breaking combination is chosen so that neither the horizontal nor the vertical spacing is too cramped or stretched.

Two types of blocks can contain layout settings: `\paper {...}` and `\layout {...}`. The `\paper` block contains page layout settings that are expected to be the same for all scores in a book or bookpart, such as the paper height, or whether to print page numbers, etc. See Sezione 4.1 [Page layout], pagina 502. The `\layout` block contains score layout settings, such as the number of systems to use, or the space between staff-groups, etc. See Sezione 4.2 [Score layout], pagina 512.

4.1 Page layout

This section discusses page layout options for the `\paper` block.

4.1.1 The \paper block

`\paper` blocks may be placed in three different places to form a descending hierarchy of `\paper` blocks:

- At the top of the input file, before all `\book`, `\bookpart`, and `\score` blocks.
- Within a `\book` block but outside all the `\bookpart` and `\score` blocks within that book.
- Within a `\bookpart` block but outside all `\score` blocks within that bookpart.

A `\paper` block cannot be placed within a `\score` block.

The values of the fields filter down this hierarchy, with the values set higher in the hierarchy persisting unless they are over-ridden by a value set lower in the hierarchy.

Several `\paper` blocks can appear at each of the levels, for example as parts of several `\included` files. If so, the fields at each level are merged, with values encountered last taking precedence if duplicated fields appear.

Settings that can appear in a `\paper` block include:

- the `set-paper-size` scheme function,
- `\paper` variables used for customizing page layout, and
- markup definitions used for customizing the layout of headers, footers, and titles.

The `set-paper-size` function is discussed in the next section, Sezione 4.1.2 [Paper size and automatic scaling], pagina 503. The `\paper` variables that deal with page layout are discussed in later sections. The markup definitions that deal with headers, footers, and titles are discussed in Sezione 3.2.2 [Custom titles headers and footers], pagina 463.

Most `\paper` variables will only work in a `\paper` block. The few that will also work in a `\layout` block are listed in Sezione 4.2.1 [The \layout block], pagina 512.

Except when specified otherwise, all `\paper` variables that correspond to distances on the page are measured in millimeters, unless a different unit is specified by the user. For example, the following declaration sets `top-margin` to ten millimeters:

```
\paper {
  top-margin = 10
}
```

To set it to `0.5` inches, use the `\in` unit suffix:

```
\paper {
  top-margin = 0.5\in
}
```

The available unit suffixes are `\mm`, `\cm`, `\in`, and `\pt`. These units are simple values for converting from millimeters; they are defined in '`ly/paper-defaults-init.ly`'. For the sake of clarity, when using millimeters, the `\mm` is typically included in the code, even though it is not technically necessary.

It is also possible to define `\paper` values using Scheme. The Scheme equivalent of the above example is:

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

Vedi anche

Notation Reference: Sezione 4.1.2 [Paper size and automatic scaling], pagina 503, Sezione 3.2.2 [Custom titles headers and footers], pagina 463, Sezione 4.2.1 [The `\layout` block], pagina 512.

Installed Files: '`ly/paper-defaults-init.ly`'.

4.1.2 Paper size and automatic scaling

Setting the paper size

'A4' is the default value when no explicit paper size is set. However, there are two functions that can be used to change it:

`set-default-paper-size`

```
#(set-default-paper-size "quarto")
```

which must always be placed at the toplevel scope, and

`set-paper-size`

```
\paper {
  #(set-paper-size "tabloid")
}
```

which must always be placed in a `\paper` block.

If the `set-default-paper-size` function is used in the toplevel scope, it must come before any `\paper` block. `set-default-paper-size` sets the paper size for all pages, whereas `set-paper-size` only sets the paper size for the pages that the `\paper` block applies to. For example, if the `\paper` block is at the top of the file, then it will apply the paper size to all pages. If the `\paper` block is inside a `\book`, then the paper size will only apply to that book.

When the `set-paper-size` function is used, it must be placed *before* any other functions used within the same `\paper` block. See [Automatic scaling to paper size], pagina 504.

Paper sizes are defined in '`scm/paper.scm`', and while it is possible to add custom sizes, they will be overwritten on subsequent software updates. The available paper sizes are listed in Sezione A.5 [Predefined paper sizes], pagina 626.

The following command can be used in the file to add a custom paper size which can then be used with `set-default-paper-size` or `set-paper-size` as appropriate,

```
#(set! paper-alist (cons '("my size" . (cons (* 15 in) (* 3 in))) paper-alist))

\paper {
  #(set-paper-size "my size")
}
```

The units `in` (inches), `cm` (centimeters) and `mm` (millimeters) can all be used.

If the symbol `'landscape` is added to the paper size function, pages will be rotated by 90 degrees, and wider line widths will be set accordingly.

```
#(set-default-paper-size "a6" 'landscape)
```

Swapping the paper dimensions *without* having the print rotated (like when printing to postcard size, or creating graphics for inclusion rather than a standalone document) can be achieved by appending '`landscape`' to the name of the paper size itself:

```
#(set-default-paper-size "a6landscape")
```

When the paper size ends with an explicit '`landscape`' or '`portrait`', the presence of a `'landscape` symbol *only* affects print orientation, not the paper dimensions used for layout.

Vedi anche

Notation Reference: [Automatic scaling to paper size], pagina 504, Sezione A.5 [Predefined paper sizes], pagina 626.

Installed Files: '`scm/paper.scm`'.

Automatic scaling to paper size

If the paper size is changed with one of the scheme functions (`set-default-paper-size` or `set-paper-size`), the values of several `\paper` variables are automatically scaled to the new size. To bypass the automatic scaling for a particular variable, set the variable after setting the paper size. Note that the automatic scaling is not triggered by setting the `paper-height` or `paper-width` variables, even though `paper-width` can influence other values (this is separate from scaling and is discussed below). The `set-default-paper-size` and `set-paper-size` functions are described in [Setting the paper size], pagina 503.

The vertical dimensions affected by automatic scaling are `top-margin` and `bottom-margin` (see Sezione 4.1.3 [Fixed vertical spacing \paper variables], pagina 504). The horizontal dimensions affected by automatic scaling are `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent`, and `short-indent` (see Sezione 4.1.5 [Horizontal spacing \paper variables], pagina 507).

The default values for these dimensions are set in '`ly/paper-defaults-init.ly`', using internal variables named `top-margin-default`, `bottom-margin-default`, etc. These are the values that result at the default paper size `a4`. For reference, with `a4` paper the `paper-height` is `297\mm` and the `paper-width` is `210\mm`.

Vedi anche

Notation Reference: Sezione 4.1.3 [Fixed vertical spacing \paper variables], pagina 504, Sezione 4.1.5 [Horizontal spacing \paper variables], pagina 507.

Installed Files: '`ly/paper-defaults-init.ly`', '`scm/paper.scm`'.

4.1.3 Fixed vertical spacing \paper variables

Nota: Some `\paper` dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], pagina 504.

Default values (before scaling) are defined in '`ly/paper-defaults-init.ly`'.

`paper-height`
: The height of the page, unset by default. Note that the automatic scaling of some vertical dimensions is not affected by this.

`top-margin`
: The margin between the top of the page and the top of the printable area. If the paper size is modified, this dimension's default value is scaled accordingly.

`bottom-margin`
: The margin between the bottom of the printable area and the bottom of the page. If the paper size is modified, this dimension's default value is scaled accordingly.

`ragged-bottom`
: If this is set to true, systems will be set at their natural spacing, neither compressed nor stretched vertically to fit the page.

`ragged-last-bottom`
: If this is set to false, then the last page, and the last page in each section created with a `\bookpart` block, will be vertically justified in the same way as the earlier pages.

Vedi anche

Notation Reference: [Automatic scaling to paper size], pagina 504.

Installed Files: '`ly/paper-defaults-init.ly`'.

Snippets: Sezione "Spacing" in *Frammenti di codice*.

Problemi noti e avvertimenti

The titles (from the `\header` block) are treated as a system, so `ragged-bottom` and `ragged-last-bottom` will add space between the titles and the first system of the score.

Explicitly defined paper-sizes will override any user-defined top or bottom margin settings.

4.1.4 Flexible vertical spacing `\paper` variables

In most cases, it is preferable for the vertical distances between certain items (such as margins, titles, systems, and separate scores) to be flexible, so that they stretch and compress nicely according to each situation. A number of `\paper` variables (listed below) are available to fine-tune the stretching behavior of these dimensions.

Note that the `\paper` variables discussed in this section do not control the spacing of staves within individual systems. Within-system spacing is controlled by grob properties, with settings typically entered inside a `\score` or `\layout` block, and not inside a `\paper` block. See Sezione 4.4.1 [Flexible vertical spacing within systems], pagina 523.

Structure of flexible vertical spacing alists

Each of the flexible vertical spacing `\paper` variables is an alist (association list) containing four *keys*:

- `basic-distance` – the vertical distance, measured in staff-spaces, between the *reference points* of the two items, when no collisions would result, and no stretching or compressing is in effect. The reference point of a (title or top-level) markup is its highest point, and the reference point of a system is the vertical center of the nearest `StaffSymbol` – even if a non-staff line (such as a `Lyrics` context) is in the way. Values for `basic-distance` that are less than either `padding` or `minimum-distance` are not meaningful, since the resulting distance will never be less than either `padding` or `minimum-distance`.

- `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect. Values for `minimum-distance` that are less than `padding` are not meaningful, since the resulting distance will never be less than `padding`.
- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result). When positive, the significance of a particular dimension's `stretchability` value lies only in its relation to the `stretchability` values of the other dimensions. For example, if one dimension has twice the `stretchability` of another, it will stretch twice as easily. Values should be non-negative and finite. The value `+inf.0` triggers a `programming_error` and is ignored, but `1.0e7` can be used for an almost infinitely stretchable spring. If unset, the default value is set to `basic-distance`. Note that the dimension's propensity to *compress* cannot be directly set by the user and is equal to (`basic-distance` − `minimum-distance`).

If a page has a ragged bottom, the resulting distance is the largest of:

- `basic-distance`,
- `minimum-distance`, and
- `padding` plus the smallest distance necessary to eliminate collisions.

For multi-page scores with a ragged bottom on the last page, the last page uses the same spacing as the preceding page, provided there is enough space for that.

Specific methods for modifying alists are discussed in Sezione 5.3.6 [Modifying alists], pagina 581. The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely redefines the variable:

```
\paper {
  system-system-spacing #'basic-distance = #8
  score-system-spacing =
    #'((basic-distance . 12)
       (minimum-distance . 6)
       (padding . 1)
       (stretchability . 12))
}
```

List of flexible vertical spacing \paper variables

The names of these variables follow the format *`upper-lower-`*`spacing`, where *`upper`* and *`lower`* are the items to be spaced. Each distance is measured between the reference points of the two items (see the description of the alist structure above). Note that in these variable names, the term '`markup`' refers to both *title markups* (`bookTitleMarkup` or `scoreTitleMarkup`) and *top-level markups* (see Sezione 3.1.5 [File structure], pagina 454). All distances are measured in staff-spaces.

Default settings are defined in '`ly/paper-defaults-init.ly`'.

`markup-system-spacing`
: the distance between a (title or top-level) markup and the system that follows it.

`score-markup-spacing`
: the distance between the last system of a score and the (title or top-level) markup that follows it.

`score-system-spacing`
: the distance between the last system of a score and the first system of the score that follows it, when no (title or top-level) markup exists between them.

`system-system-spacing`
: the distance between two systems in the same score.

`markup-markup-spacing`
: the distance between two (title or top-level) markups.

`last-bottom-spacing`
: the distance from the last system or top-level markup on a page to the bottom of the printable area (i.e. the top of the bottom margin).

`top-system-spacing`
: the distance from the top of the printable area (i.e. the bottom of the top margin) to the first system on a page, when there is no (title or top-level) markup between the two.

`top-markup-spacing`
: the distance from the top of the printable area (i.e. the bottom of the top margin) to the first (title or top-level) markup on a page, when there is no system between the two.

Vedi anche

Notation Reference: Sezione 4.4.1 [Flexible vertical spacing within systems], pagina 523.

Installed Files: '`ly/paper-defaults-init.ly`'.

Snippets: Sezione "Spacing" in *Frammenti di codice*.

4.1.5 Horizontal spacing \paper variables

Nota: Some `\paper` dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], pagina 504.

\paper variables for widths and margins

Default values (before scaling) that are not listed here are defined in '`ly/paper-defaults-init.ly`'.

`paper-width`
: The width of the page, unset by default. While `paper-width` has no effect on the automatic scaling of some horizontal dimensions, it does influence the `line-width` variable. If both `paper-width` and `line-width` are set, then `left-margin` and `right-margin` will also be updated. Also see `check-consistency`.

`line-width`
: The horizontal extent of the staff lines in unindented, non-ragged systems, equal to `(paper-width − left-margin − right-margin)` when unset. If `line-width` is set, and both `left-margin` and `right-margin` are unset, then the margins will be updated to center the systems on the page automatically. Also see `check-consistency`. This variable can also be set in a `\layout` block.

`left-margin`
: The margin between the left edge of the page and the start of the staff lines in unindented systems. If the paper size is modified, this dimension's default value is scaled accordingly. If `left-margin` is unset, and both `line-width` and `right-margin` are set, then `left-margin` is set to `(paper-width − line-width − right-margin)`. If only `line-width` is set, then both margins are set to `((paper-width − line-width) / 2)`, and the systems are consequently centered on the page. Also see `check-consistency`.

`right-margin`
: The margin between the right edge of the page and the end of the staff lines in non-ragged systems. If the paper size is modified, this dimension's default value is scaled accordingly. If `right-margin` is unset, and both `line-width` and `left-margin` are set, then `right-margin` is set to `(paper-width − line-width − left-margin)`. If only `line-width` is set, then both margins are set to `((paper-width − line-width) / 2)`, and the systems are consequently centered on the page. Also see `check-consistency`.

`check-consistency`
: If set to true, print a warning if `left-margin`, `line-width`, and `right-margin` do not exactly add up to `paper-width`, and replace each of these (except `paper-width`) with its default value (scaled to the paper size if necessary). If set to false, ignore any inconsistencies and allow systems to run off the edge of the page.

`ragged-right`
: If set to true, systems will not fill the line width. Instead, systems end at their natural horizontal length. Default: `#t` for scores with only one system, and `#f` for scores with two or more systems. This variable can also be set in a `\layout` block.

`ragged-last`
: If set to true, the last system in the score will not fill the line width. Instead the last system ends at its natural horizontal length. Default: `#f`. This variable can also be set in a `\layout` block.

Vedi anche

Notation Reference: [Automatic scaling to paper size], pagina 504.

Installed Files: '`ly/paper-defaults-init.ly`'.

Problemi noti e avvertimenti

Explicitly defined paper-sizes will override any user-defined left or right margin settings.

\paper variables for two-sided mode

Default values (before scaling) are defined in '`ly/paper-defaults-init.ly`'.

`two-sided`
: If set to true, use `inner-margin`, `outer-margin` and `binding-offset` to determine margins depending on whether the page number is odd or even. This overrides `left-margin` and `right-margin`.

`inner-margin`
: The margin all pages have at the inner side if they are part of a book. If the paper size is modified, this dimension's default value is scaled accordingly. Works only with `two-sided` set to true.

`outer-margin`
: The margin all pages have at the outer side if they are part of a book. If the paper size is modified, this dimension's default value is scaled accordingly. Works only with `two-sided` set to true.

`binding-offset`
: The amount `inner-margin` is increased to make sure nothing will be hidden by the binding. If the paper size is modified, this dimension's default value is scaled accordingly. Works only with `two-sided` set to true.

Vedi anche

Notation Reference: [Automatic scaling to paper size], pagina 504.

Installed Files: ‘`ly/paper-defaults-init.ly`’.

\paper variables for shifts and indents

Default values (before scaling) that are not listed here are defined in ‘`ly/paper-defaults-init.ly`’.

`horizontal-shift`
: The amount that all systems (including titles and system separators) are shifted to the right. Default: `0.0\mm`.

`indent`
: The level of indentation for the first system in a score. If the paper size is modified, this dimension's default value is scaled accordingly. This variable can also be set in a `\layout` block.

`short-indent`
: The level of indentation for all systems in a score besides the first system. If the paper size is modified, this dimension's default value is scaled accordingly. This variable can also be set in a `\layout` block.

Vedi anche

Notation Reference: [Automatic scaling to paper size], pagina 504.

Installed Files: ‘`ly/paper-defaults-init.ly`’.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.1.6 Other \paper variables

\paper variables for line breaking

`max-systems-per-page`
: The maximum number of systems that will be placed on a page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`min-systems-per-page`
: The minimum number of systems that will be placed on a page. This may cause pages to be overfilled if it is made too large. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`systems-per-page`
: The number of systems that should be placed on each page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`system-count`
: The number of systems to be used for a score. Default: unset. This variable can also be set in a `\layout` block.

Vedi anche

Notation Reference: Sezione 4.3.1 [Line breaking], pagina 515.

\paper variables for page breaking

Default values not listed here are defined in '`ly/paper-defaults-init.ly`'

`page-breaking`
: The page-breaking algorithm to use. Choices are `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-line-breaking` and `ly:optimal-breaking` (the default).

`page-breaking-system-system-spacing`
: Tricks the page breaker into thinking that `system-system-spacing` is set to something different than it really is. For example, if `page-breaking-system-system-spacing #'padding` is set to something substantially larger than `system-system-spacing #'padding`, then the page-breaker will put fewer systems on each page. Default: unset.

`page-count`
: The number of pages to be used for a score, unset by default.

The following variables are effective only when `page-breaking` is set to `ly:page-turn-breaking`. Page breaks are then chosen to minimize the number of page turns. Since page turns are required on moving from an odd-numbered page to an even-numbered one, a layout in which the last page is odd-numbered will usually be favoured. Places where page turns are preferred can be indicated manually by inserting `\allowPageTurn` or automatically by including the `Page_turn_engraver` (see Sezione 4.3.4 [Optimal page turning], pagina 518).

If there are insufficient choices available for making suitable page turns, LilyPond may insert a blank page either within a score, between scores (if there are two or more scores), or by ending a score on an even-numbered page. The values of the following three variables may be increased to make these actions less likely.

The values are penalties, i.e. the higher the value the less likely will be the associated action relative to other choices.

`blank-page-penalty`
: The penalty for having a blank page in the middle of a score. If `blank-page-penalty` is large and `ly:page-turn-breaking` is selected, then LilyPond will be less likely to insert a page in the middle of a score. Instead, it will space out the music further to fill the blank page and the following one. Default: 5.

`blank-last-page-penalty`
: The penalty for ending the score on an even-numbered page. If `blank-last-page-penalty` is large and `ly:page-turn-breaking` is selected, then LilyPond will be less likely to produce a score in which the last page is even-numbered. Instead, it will adjust the spacing in order to use one page more or one page less. Default: 0.

`blank-after-score-page-penalty`
: The penalty for having a blank page after the end of one score and before the next. By default, this is smaller than `blank-page-penalty`, so that blank pages after scores are inserted in preference to blank pages within a score. Default: 2.

Vedi anche

Notation Reference: Sezione 4.3.2 [Page breaking], pagina 517, Sezione 4.3.3 [Optimal page breaking], pagina 518, Sezione 4.3.4 [Optimal page turning], pagina 518, Sezione 4.3.5 [Minimal page breaking], pagina 519, Sezione 4.3.6 [One-line page breaking], pagina 519.

Installed Files: '`ly/paper-defaults-init.ly`'.

\paper variables for page numbering

Default values not listed here are defined in '`ly/paper-defaults-init.ly`'

`auto-first-page-number`
: The page breaking algorithm is affected by the first page number being odd or even. If set to true, the page breaking algorithm will decide whether to start with an odd or even number. This will result in the first page number remaining as is or being increased by one. Default: `#f`.

`first-page-number`
: The value of the page number on the first page.

`print-first-page-number`
: If set to true, a page number is printed on the first page.

`print-page-number`
: If set to false, page numbers are not printed.

Vedi anche

Installed Files: '`ly/paper-defaults-init.ly`'.

Problemi noti e avvertimenti

Odd page numbers are always on the right. If you want the music to start on page 1 there must be a blank page on the back of the cover page so that page 1 is on the right hand side.

Miscellaneous \paper variables

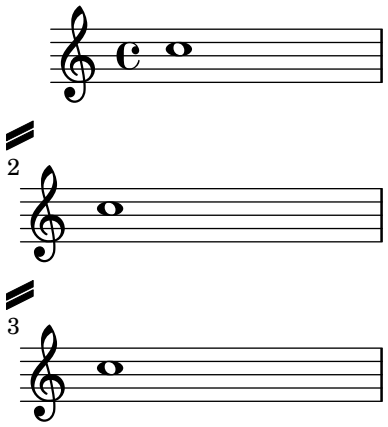
`page-spacing-weight`
: The relative importance of page (vertical) spacing and line (horizontal) spacing. High values will make page spacing more important. Default: `10`.

`print-all-headers`
: If set to true, this will print all headers for each `\score` in the output. Normally only the `piece` and `opus` header variables are printed. Default: `#f`.

`system-separator-markup`
: A markup object that is inserted between systems, often used for orchestral scores. Default: unset. The `\slashSeparator` markup, defined in '`ly/titling-init.ly`', is provided as a sensible default, for example:

```
#(set-default-paper-size "a8")

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative c'' { c1 \break c1 \break c1 }
  }
}
```



Vedi anche

Installed Files: '`ly/titling-init.ly`'.

Snippets: Sezione "Spacing" in *Frammenti di codice*.

Problemi noti e avvertimenti

The default page header puts the page number and the `instrument` field from the `\header` block on a line.

4.2 Score layout

This section discusses score layout options for the `\layout` block.

4.2.1 The `\layout` block

While the `\paper` block contains settings that relate to the page formatting of the whole document, the `\layout` block contains settings for score-specific layout. To set score layout options globally, enter them in a toplevel `\layout` block. To set layout options for an individual score, enter them in a `\layout` block inside the `\score` block, after the music. Settings that can appear in a `\layout` block include:

- the `layout-set-staff-size` scheme function,
- context modifications in `\context` blocks, and
- `\paper` variables that affect score layout.

The `layout-set-staff-size` function is discussed in the next section, Sezione 4.2.2 [Setting the staff size], pagina 514. Context modifications are discussed in a separate chapter; see Sezione 5.1.4 [Modifying context plug-ins], pagina 561 and Sezione 5.1.5 [Changing context default settings], pagina 563. The `\paper` variables that can appear in a `\layout` block are:

- `line-width`, `ragged-right` and `ragged-last` (see [`\paper` variables for widths and margins], pagina 507)
- `indent` and `short-indent` (see [`\paper` variables for shifts and indents], pagina 509)
- `system-count` (see [`\paper` variables for line breaking], pagina 509)

Here is an example `\layout` block:

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGrouper.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
```

```
    \override TextScript.padding = #1
    \override Glissando.thickness = #3
  }
}
```

Multiple `\layout` blocks can be entered as toplevel expressions. This can, for example, be useful if different settings are stored in separate files and included optionally. Internally, when a `\layout` block is evaluated, a copy of the current `\layout` configuration is made, then any changes defined within the block are applied and the result is saved as the new current configuration. From the user's perspective the `\layout` blocks are combined, but in conflicting situations (when the same property is changed in different blocks) the later definitions take precedence.

For example, if this block:

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = #1.5
  }
}
```

is placed after the one from the preceding example the `'padding` and `'color` overrides for `TextScript` are combined, but the later `'thickness` override for `Glissando` replaces (or hides) the earlier one.

`\layout` blocks may be assigned to variables for reuse later, but the way this works is slightly but significantly different from writing them literally.

If a variable is defined like this:

```
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = #4
  }
}
```

it will hold the current `\layout` configuration with the `NoteHead.font-size` override added, but this combination is *not* saved as the new current configuration. Be aware that the 'current configuration' is read when the variable is defined and not when it is used, so the content of the variable is dependent on its position in the source.

The variable can then be used inside another `\layout` block, for example:

```
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

A `\layout` block containing a variable, as in the example above, does *not* copy the current configuration but instead uses the content of `\layoutVariable` as the base configuration for the further additions. This means that any changes defined between the definition and the use of the variable are lost.

If `layoutVariable` is defined (or `\included`) immediately before being used, its content is just the current configuration plus the overrides defined within it. So in the example above showing the use of `\layoutVariable` the final `\layout` block would consist of:

```
  TextScript.padding = #1
  TextScript.color = #magenta
  Glissando.thickness = #1.5
  NoteHead.font-size = #4
  NoteHead.color = #red
```

plus the `indent` and the `StaffGrouper` overrides.

But if the variable had already been defined before the first `\layout` block the current configuration would now contain only

```
  NoteHead.font-size = #4 % (written in the variable definition)
  NoteHead.color = #red % (added after the use of the variable)
```

If carefully planned, `\layout` variables can be a valuable tool to structure the layout design of sources, and also to reset the `\layout` configuration to a known state.

Vedi anche

Notation Reference: Sezione 5.1.5 [Changing context default settings], pagina 563.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.2.2 Setting the staff size

The default **staff size** is set to 20 points. This may be changed in two ways:

To set the staff size globally for all scores in a file (or in a `book` block, to be precise), use `set-global-staff-size`.

```
#(set-global-staff-size 14)
```

This sets the global default size to 14pt staff height and scales all fonts accordingly.

To set the staff size individually for each score, use

```
\score{
  ...
  \layout {
    #(layout-set-staff-size 15)
  }
}
```

The Feta font provides musical symbols at eight different sizes. Each font is tuned for a different staff size: at a smaller size the font becomes heavier, to match the relatively heavier staff lines. The recommended font sizes are listed in the following table:

font name	staff height (pt)	staff height (mm)	use
feta11	11.22	3.9	pocket scores
feta13	12.60	4.4	
feta14	14.14	5.0	
feta16	15.87	5.6	
feta18	17.82	6.3	song books
feta20	20	7.0	standard parts
feta23	22.45	7.9	
feta26	25.2	8.9	

These fonts are available in any sizes. The context property `fontSize` and the layout property `staff-space` (in Sezione “StaffSymbol” in *Guida al Funzionamento Interno*) can be used to tune the size for individual staves. The sizes of individual staves are relative to the global size.

Vedi anche

Notation Reference: ⟨undefined⟩ [Selecting notation font size], pagina ⟨undefined⟩.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

Problemi noti e avvertimenti

`layout-set-staff-size` does not change the distance between the staff lines.
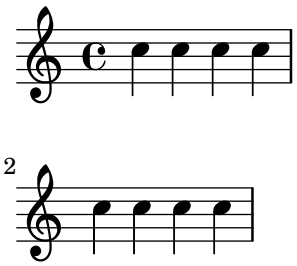
4.3 Breaks

4.3.1 Line breaking

Line breaks are normally determined automatically. They are chosen so that lines look neither cramped nor loose, and consecutive lines have similar density.
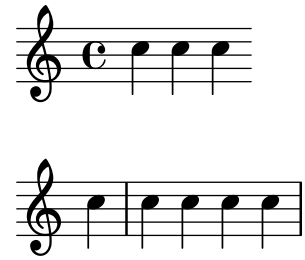
To manually force a line break at a bar line, use the `\break` command:

```
c4 c c c | \break
c4 c c c |
```



By default, a `\break` in the middle of a measure is ignored, and a warning is printed. To force a line break in the middle of a measure, add an invisible bar line with ‘`\bar ""`’:

```
c4 c c
\bar "" \break
c |
c4 c c c |
```
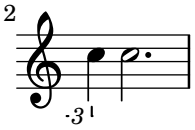


A `\break` occurring at a bar line is also ignored if the previous measure ends in the middle of a note, such as when a tuplet begins and ends in different measures. To allow `\break` commands to work in these situations, remove the `Forbid_line_break_engraver` from the `Voice` context. Note that manually forced line breaks have to be added in parallel with the music:

```
\new Voice \with {
  \remove "Forbid_line_break_engraver"
} \relative c'' {
  <<
    { c2. \tuplet 3/2 { c4 c c } c2. | }
    { s1 | \break s1 | }
```
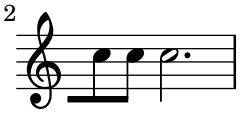
```
  >>
}
```





Similarly, line breaks are normally forbidden when beams cross bar lines. This behavior can be changed by setting `\override Beam.breakable = ##t`:

```
\override Beam.breakable = ##t
c2. c8[ c | \break
c8 c] c2. |
```





The `\noBreak` command forbids a line break at the bar line where it is inserted.

The most basic settings influencing line spacing are `indent` and `line-width`. They are set in the `\layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `\layout` block, then systems end at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is.

The option `ragged-last` is similar to `ragged-right`, but affects only the last line of the piece.

```
\layout {
  indent = 0\mm
  line-width = 150\mm
  ragged-last = ##t
}
```

For line breaks at regular intervals use `\break` separated by skips and repeated with `\repeat`. For example, this would cause the following 28 measures (assuming 4/4 time) to be broken every 4 measures, and only there:

```
<<
  \repeat unfold 7 {
    s1 \noBreak s1 \noBreak
    s1 \noBreak s1 \break
  }
  { the actual music... }
>>
```

Comandi predefiniti

`\break`, `\noBreak`.

Vedi anche

Notation Reference: [`\paper` variables for line breaking], pagina 509.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

Internals Reference: Sezione “LineBreakEvent” in *Guida al Funzionamento Interno*.

4.3.2 Page breaking

The default page breaking may be overridden by inserting `\pageBreak` or `\noPageBreak` commands. These commands are analogous to `\break` and `\noBreak`. They should be inserted at a bar line. These commands force and forbid a page-break from happening. Of course, the `\pageBreak` command also forces a line break.

The `\pageBreak` and `\noPageBreak` commands may also be inserted at top-level, between scores and top-level markups.

There are also analogous settings to `ragged-right` and `ragged-last` which have the same effect on vertical spacing. If `ragged-bottom` is set to `#t` the systems will not be justified vertically. When `ragged-last-bottom` is set to `#t`, as it is by default, empty space is allowed at the bottom of the final page (or the final page in each `\bookpart`). See Sezione 4.1.3 [Fixed vertical spacing `\paper` variables], pagina 504.

Page breaks are computed by the `page-breaking` function. LilyPond provides three algorithms for computing page breaks, `ly:optimal-breaking`, `ly:page-turn-breaking` and `ly:minimal-breaking`. The default is `ly:optimal-breaking`, but the value can be changed in the `\paper` block:

```
\paper {
  page-breaking = #ly:page-turn-breaking
}
```

When a book has many scores and pages, the page breaking problem may be difficult to solve, requiring large processing time and memory. To ease the page breaking process, `\bookpart` blocks are used to divide the book into several parts: the page breaking occurs separately on each part. Different page breaking functions may also be used in different book parts.

```
\bookpart {
  \header {
    subtitle = "Preface"
  }
  \paper {
     %% In a part consisting mostly of text,
     %% ly:minimal-breaking may be preferred
     page-breaking = #ly:minimal-breaking
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In this part, consisting of music, the default optimal
  %% page breaking function is used.
  \header {
    subtitle = "First movement"
  }
```

```
  \score { ... }
  ...
}
```

Comandi predefiniti

`\pageBreak`, `\noPageBreak`.

Vedi anche

Notation Reference: [`\paper` variables for page breaking], pagina 510.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.3.3 Optimal page breaking

The `ly:optimal-breaking` function is LilyPond's default method of determining page breaks. It attempts to find a page breaking that minimizes cramping and stretching, both horizontally and vertically. Unlike `ly:page-turn-breaking`, it has no concept of page turns.

Vedi anche

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.3.4 Optimal page turning

Often it is necessary to find a page breaking configuration so that there is a rest at the end of every second page. This way, the musician can turn the page without having to miss notes. The `ly:page-turn-breaking` function attempts to find a page breaking minimizing cramping and stretching, but with the additional restriction that it is only allowed to introduce page turns in specified places.

There are two steps to using this page breaking function. First, you must enable it in the `\paper` block, as explained in Sezione 4.3.2 [Page breaking], pagina 517. Then you must tell the function where you would like to allow page breaks.

There are two ways to achieve the second step. First, you can specify each potential page turn manually, by inserting `\allowPageTurn` into your input file at the appropriate places.

If this is too tedious, you can add a `Page_turn_engraver` to a Staff or Voice context. The `Page_turn_engraver` will scan the context for sections without notes (note that it does not scan for rests; it scans for the absence of notes. This is so that single-staff polyphony with rests in one of the parts does not throw off the `Page_turn_engraver`). When it finds a sufficiently long section without notes, the `Page_turn_engraver` will insert an `\allowPageTurn` at the final bar line in that section, unless there is a ‘special’ bar line (such as a double bar), in which case the `\allowPageTurn` will be inserted at the final ‘special’ bar line in the section.

The `Page_turn_engraver` reads the context property `minimumPageTurnLength` to determine how long a note-free section must be before a page turn is considered. The default value for `minimumPageTurnLength` is `(ly:make-moment 1/1)`. If you want to disable page turns, you can set it to something very large.

```
\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % a page turn will be allowed here
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5/2)
  R1 | % a page turn will not be allowed here
  a4 b r2 |
  R1*2 | % a page turn will be allowed here
```

```
  a1
}
```

The `Page_turn_engraver` detects volta repeats. It will only allow a page turn during the repeat if there is enough time at the beginning and end of the repeat to turn the page back. The `Page_turn_engraver` can also disable page turns if the repeat is very short. If you set the context property `minimumRepeatLengthForPageTurn` then the `Page_turn_engraver` will only allow turns in repeats whose duration is longer than this value.

The page turning commands, `\pageTurn`, `\noPageTurn` and `\allowPageTurn`, may also be used at top-level, between scores and top-level markups.

Comandi predefiniti

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

Vedi anche

Notation Reference: [`\paper` variables for line breaking], pagina 509.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

Problemi noti e avvertimenti

There should only be one `Page_turn_engraver` in a score. If there is more than one, they will interfere with each other.

4.3.5 Minimal page breaking

The `ly:minimal-breaking` function performs minimal computations to calculate the page breaking: it fills a page with as many systems as possible before moving to the next one. Thus, it may be preferred for scores with many pages, where the other page breaking functions could be too slow or memory demanding, or a lot of texts. It is enabled using:

```
\paper {
  page-breaking = #ly:minimal-breaking
}
```

Vedi anche

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.3.6 One-line page breaking

The `ly:one-line-breaking` function is a special-purpose page breaking algorithm that puts each score on its own page, and on a single line. This page breaking function does not typeset titles or margins; only the score will be displayed.

The page width will be adjusted so that the longest score fits on one line. In particular, `paper-width`, `line-width` and `indent` variables in the `\paper` block will be ignored, although `left-margin` and `right-margin` will still be honored. The height of the page will be left unmodified.

4.3.7 Explicit breaks

Lily sometimes rejects explicit `\break` and `\pageBreak` commands. There are two commands to override this behavior:

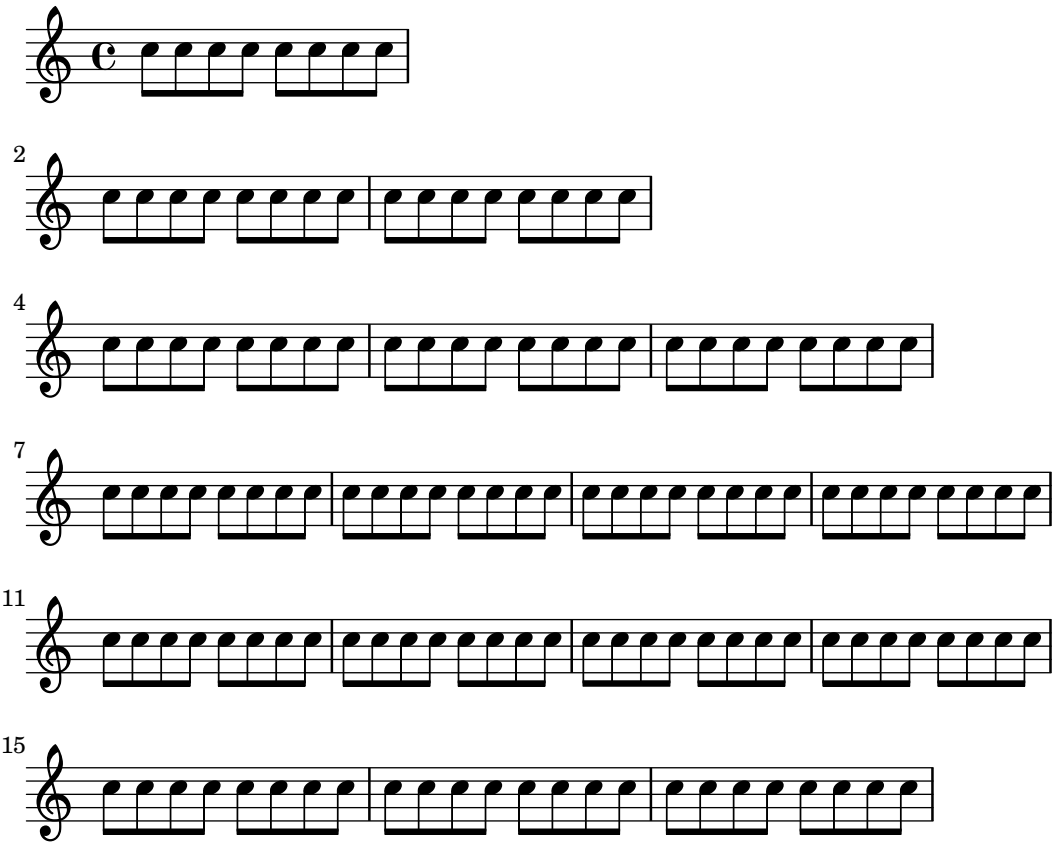
```
\override NonMusicalPaperColumn.line-break-permission = ##f
\override NonMusicalPaperColumn.page-break-permission = ##f
```

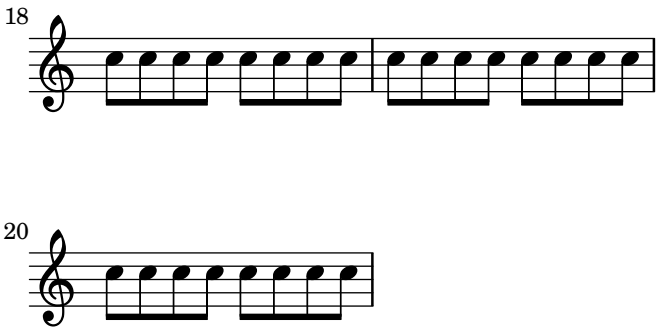
When `line-break-permission` is overridden to false, Lily will insert line breaks at explicit `\break` commands and nowhere else. When `page-break-permission` is overridden to false, Lily will insert page breaks at explicit `\pageBreak` commands and nowhere else.

```
\paper {
  indent = #0
  ragged-right = ##t
  ragged-bottom = ##t
}

music = \relative c'' { c8 c c c }

\score {
  \new Staff {
    \repeat unfold 2 { \music } \break
    \repeat unfold 4 { \music } \break
    \repeat unfold 6 { \music } \break
    \repeat unfold 8 { \music } \pageBreak
    \repeat unfold 8 { \music } \break
    \repeat unfold 6 { \music } \break
    \repeat unfold 4 { \music } \break
    \repeat unfold 2 { \music }
  }
  \layout {
    \context {
      \Score
      \override NonMusicalPaperColumn.line-break-permission = ##f
      \override NonMusicalPaperColumn.page-break-permission = ##f
    }
  }
}
```





Vedi anche

Snippets: Sezione "Spacing" in *Frammenti di codice*.

4.3.8 Using an extra voice for breaks

Line- and page-breaking information usually appears within note entry directly.
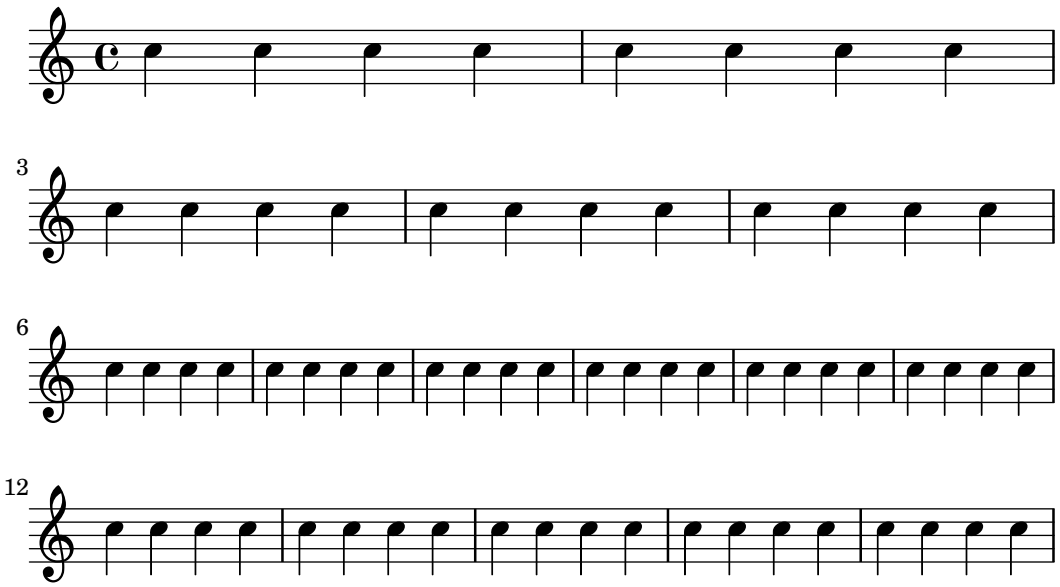
```
music = \relative c'' { c4 c c c }

\score {
  \new Staff {
    \repeat unfold 2 { \music } \break
    \repeat unfold 3 { \music }
  }
}
```

This makes `\break` and `\pageBreak` commands easy to enter but mixes music entry with information that specifies how music should lay out on the page. You can keep music entry and line- and page-breaking information in two separate places by introducing an extra voice to contain the breaks. This extra voice contains only skips together with `\break`, `pageBreak` and other breaking layout information.

```
music = \relative c'' { c4 c c c }

\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    \new Staff <<
      \new Voice {
        s1 * 2 \break
        s1 * 3 \break
        s1 * 6 \break
        s1 * 5 \break
      }
      \new Voice {
        \repeat unfold 2 { \music }
        \repeat unfold 3 { \music }
        \repeat unfold 6 { \music }
        \repeat unfold 5 { \music }
      }
    >>
  }
}
```



This pattern becomes especially helpful when overriding `line-break-system-details` and the other useful but long properties of `NonMusicalPaperColumnGrob`, as explained in Sezione 4.4 [Vertical spacing], pagina 523.

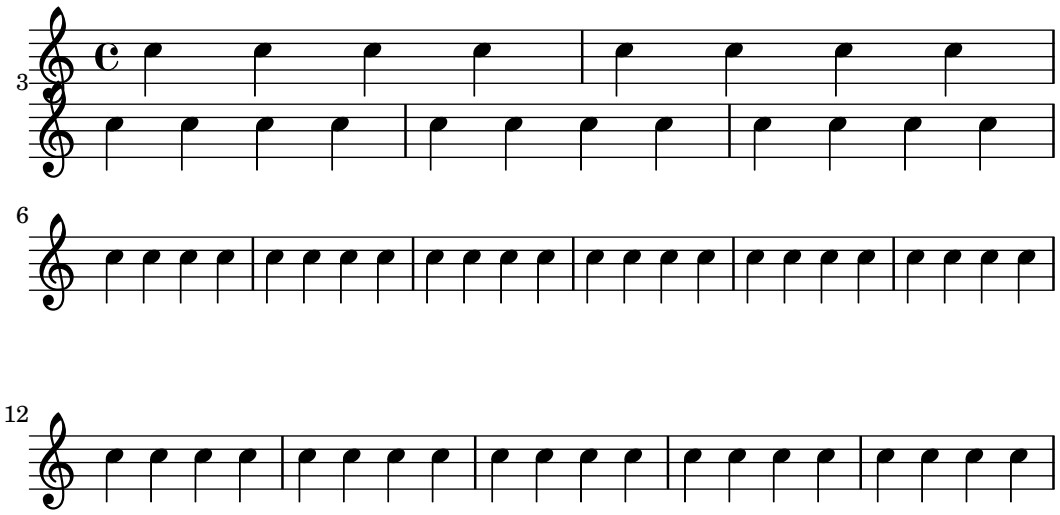
```
music = \relative c'' { c4 c c c }

\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    \new Staff <<
      \new Voice {
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 0))
        s1 * 2 \break

        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 5))
        s1 * 3 \break

        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 15))
        s1 * 6 \break

        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 30))
        s1 * 5 \break
      }
      \new Voice {
        \repeat unfold 2 { \music }
        \repeat unfold 3 { \music }
        \repeat unfold 6 { \music }
        \repeat unfold 5 { \music }
      }
    >>
  }
}
```



Vedi anche

Notation Reference: Sezione 4.4 [Vertical spacing], pagina 523.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.4 Vertical spacing

Vertical spacing is controlled by three things: the amount of space available (i.e., paper size and margins), the amount of space between systems, and the amount of space between staves inside a system.

4.4.1 Flexible vertical spacing within systems

Three separate mechanisms control the flexible vertical spacing within systems, one for each of the following categories:

- *ungrouped staves*,
- *grouped staves* (staves within a staff-group such as `ChoirStaff`, etc.), and
- *non-staff lines* (such as `Lyrics`, `ChordNames`, etc.).

The height of each system is determined in two steps. First, all of the staves are spaced according to the amount of space available. Then, the non-staff lines are distributed between the staves.

Note that the spacing mechanisms discussed in this section only control the vertical spacing of staves and non-staff lines within individual systems. The vertical spacing between separate systems, scores, markups, and margins is controlled by `\paper` variables, which are discussed in Sezione 4.1.4 [Flexible vertical spacing `\paper` variables], pagina 505.

Within-system spacing properties

The within-system vertical spacing mechanisms are controlled by two sets of grob properties. The first set is associated with the `VerticalAxisGroup` grob, which is created by all staves and non-staff lines. The second set is associated with the `StaffGrouper` grob, which can be created by staff-groups, but only if explicitly called. These properties are described individually at the end of this section.

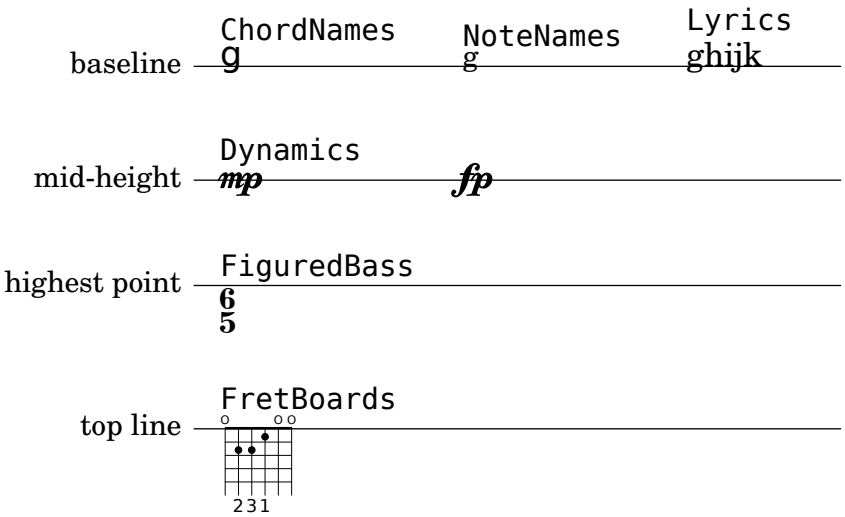
The names of these properties (except for `staff-affinity`) follow the format *`item1`*`-`*`item2`*`-spacing`, where *`item1`* and *`item2`* are the items to be spaced. Note that *`item2`* is not necessarily below *`item1`*; for example, `nonstaff-relatedstaff-spacing` will measure upwards from the non-staff line if `staff-affinity` is `UP`.

Each distance is measured between the *reference points* of the two items. The reference point for a staff is the vertical center of its `StaffSymbol` (i.e. the middle line if `line-count` is odd;

the middle space if `line-count` is even). The reference points for individual non-staff lines are given in the following table:

Non-staff line	Reference point
`ChordNames`	baseline
`NoteNames`	baseline
`Lyrics`	baseline
`Dynamics`	mid-height of 'm'
`FiguredBass`	highest point
`FretBoards`	top line

In the following image, horizontal lines indicate the positions of these reference points:



Each of the vertical spacing grob properties (except `staff-affinity`) uses the same alist structure as the `\paper` spacing variables discussed in Sezione 4.1.4 [Flexible vertical spacing \paper variables], pagina 505. Specific methods for modifying alists are discussed in Sezione 5.3.6 [Modifying alists], pagina 581. Grob properties should be adjusted with an `\override` inside a `\score` or `\layout` block, and not inside a `\paper` block.

The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely re-defines the property:

```
\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing =
    #'((basic-distance . 10)
       (minimum-distance . 9)
       (padding . 1)
       (stretchability . 10))
} { ... }
```

To change any spacing settings globally, put them in the `\layout` block:

```
\layout {
  \context {
    \Staff
```

```
    \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
  }
}
```

Standard settings for the vertical spacing grob properties are listed in Sezione "VerticalAxisGroup" in *Guida al Funzionamento Interno* and Sezione "StaffGrouper" in *Guida al Funzionamento Interno*. Default overrides for specific types of non-staff lines are listed in the relevant context descriptions in Sezione "Contexts" in *Guida al Funzionamento Interno*.

Properties of the `VerticalAxisGroup` grob

`VerticalAxisGroup` properties are typically adjusted with an `\override` at the `Staff` level (or equivalent).

`staff-staff-spacing`
: Used to determine the distance between the current staff and the staff just below it in the same system, even if one or more non-staff lines (such as `Lyrics`) are placed between the two staves. Does not apply to the bottom staff of a system.

  Initially, the `staff-staff-spacing` of a `VerticalAxisGroup` is a Scheme function that applies the properties of the `StaffGrouper` if the staff is part of a group, or the `default-staff-staff-spacing` of the staff otherwise. This allows staves to be spaced differently when they are grouped. For uniform spacing regardless of grouping, this function may be replaced by a flexible-spacing alist, using the complete-redefinition form of override shown above.

`default-staff-staff-spacing`
: A flexible-spacing alist defining the `staff-staff-spacing` used for ungrouped staves, unless `staff-staff-spacing` has been explicitly set with an `\override`.

`staff-affinity`
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Adjacent non-staff lines should have non-increasing `staff-affinity` from top to bottom, e.g. a non-staff line set to `UP` should not immediately follow one that is set to `DOWN`. Non-staff lines at the top of a system should use `DOWN`; those at the bottom should use `UP`. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff. Setting `staff-affinity` to `UP`, `CENTER`, or `DOWN` causes a staff to be spaced as a non-staff line.

`nonstaff-relatedstaff-spacing`
: The distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. This means that the placement of a non-staff line depends on both the surrounding staves and the surrounding non-staff lines. Setting the `stretchability` of one of these types of spacing to a small value will make that spacing dominate. Setting the `stretchability` to a large value will make that spacing have little effect.

`nonstaff-nonstaff-spacing`
: The distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`.

`nonstaff-unrelatedstaff-spacing`
: The distance between the current non-staff line and the staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. This can be used, for example, to require a minimum amount of padding between a `Lyrics` line and the staff to which it does not belong.

Properties of the `StaffGrouper` grob

`StaffGrouper` properties are typically adjusted with an `\override` at the `StaffGroup` level (or equivalent).

`staff-staff-spacing`
: The distance between consecutive staves within the current staff-group. The `staff-staff-spacing` property of an individual staff's `VerticalAxisGroup` grob can be overriden with different spacing settings for that staff.

`staffgroup-staff-spacing`
: The distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines (such as `Lyrics`) exist between the two staves. Does not apply to the bottom staff of a system. The `staff-staff-spacing` property of an individual staff's `VerticalAxisGroup` grob can be overriden with different spacing settings for that staff.

Vedi anche

Notation Reference: Sezione 4.1.4 [Flexible vertical spacing `\paper` variables], pagina 505, Sezione 5.3.6 [Modifying alists], pagina 581.

Installed Files: '`ly/engraver-init.ly`', '`scm/define-grobs.scm`'.

Internals Reference: Sezione "Contexts" in *Guida al Funzionamento Interno*, Sezione "VerticalAxisGroup" in *Guida al Funzionamento Interno*, Sezione "StaffGrouper" in *Guida al Funzionamento Interno*.

Spacing of ungrouped staves

Staves (such as `Staff`, `DrumStaff`, `TabStaff`, etc.) are contexts that can contain one or more voice contexts, but cannot contain any other staves.

The following properties affect the spacing of *ungrouped* staves:

- `VerticalAxisGroup` properties:
  - `default-staff-staff-spacing`
  - `staff-staff-spacing`

These grob properties are described individually above; see [Within-system spacing properties], pagina 523.

Additional properties are involved for staves that are part of a staff-group; see [Spacing of grouped staves], pagina 527.

The following example shows how the `default-staff-staff-spacing` property can affect the spacing of ungrouped staves. The same overrides applied to `staff-staff-spacing` would have the same effect, but would also apply in cases where the staves are combined in a group or groups.

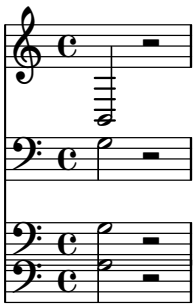
```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing =
```

```
      #'((basic-distance . 8)
         (minimum-distance . 7)
         (padding . 1))
  }
}

<<
  % The very low note here needs more room than 'basic-distance
  % can provide, so the distance between this staff and the next
  % is determined by 'padding.
  \new Staff { b,2 r | }

  % Here, 'basic-distance provides enough room, and there is no
  % need to compress the space (towards 'minimum-distance) to make
  % room for anything else on the page, so the distance between
  % this staff and the next is determined by 'basic-distance.
  \new Staff { \clef bass g2 r | }

  % By setting 'padding to a negative value, staves can be made to
  % collide.  The lowest acceptable value for 'basic-distance is 0.
  \new Staff \with {
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 3.5)
         (padding . -10))
  } { \clef bass g2 r | }
  \new Staff { \clef bass g2 r | }
>>
```



Vedi anche

Installed Files: ‘`scm/define-grobs.scm`’.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

Internals Reference: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*.

Spacing of grouped staves

In orchestral and other large scores, it is common to place staves in groups. The space between groups is typically larger than the space between staves of the same group.

Staff-groups (such as `StaffGroup`, `ChoirStaff`, etc.) are contexts that can contain one or more staves simultaneously.

The following properties affect the spacing of staves inside staff-groups:

- `VerticalAxisGroup` properties:

    - `staff-staff-spacing`
- `StaffGrouper` properties:
    - `staff-staff-spacing`
    - `staffgroup-staff-spacing`

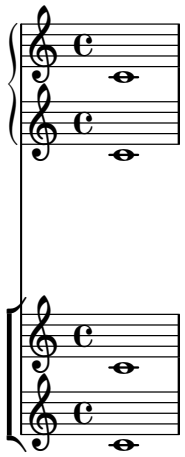
These grob properties are described individually above; see [Within-system spacing properties], pagina 523.

The following example shows how properties of the `StaffGrouper` grob can affect the spacing of grouped staves:

```
\layout {
  \context {
    \Score
    \override StaffGrouper.staff-staff-spacing.padding = #0
    \override StaffGrouper.staff-staff-spacing.basic-distance = #1
  }
}

<<
  \new PianoStaff \with {
    \override StaffGrouper.staffgroup-staff-spacing.basic-distance = #20
  } <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>

  \new StaffGroup <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>
>>
```



Vedi anche

Installed Files: ‘`scm/define-grobs.scm`’.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

Internals Reference: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “StaffGrouper” in *Guida al Funzionamento Interno*.

Spacing of non-staff lines

Non-staff lines (such as `Lyrics`, `ChordNames`, etc.) are contexts whose layout objects are engraved like staves (i.e. in horizontal lines within systems). Specifically, non-staff lines are non-staff contexts that create the `VerticalAxisGroup` layout object.

The following properties affect the spacing of non-staff lines:

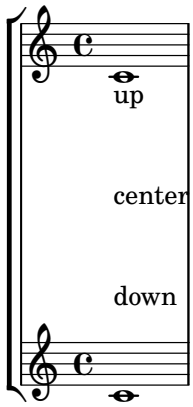
- `VerticalAxisGroup` properties:
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`
  - `nonstaff-unrelatedstaff-spacing`

These grob properties are described individually above; see [Within-system spacing properties], pagina 523.

The following example shows how the `nonstaff-nonstaff-spacing` property can affect the spacing of consecutive non-staff lines. Here, by setting the `stretchability` key to a very high value, the lyrics are able to stretch much more than usual:

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup.nonstaff-nonstaff-spacing.stretchability = #1000
  }
}

\new StaffGroup
<<
  \new Staff \with {
    \override VerticalAxisGroup.staff-staff-spacing = #'((basic-distance . 30))
  } { c'1 }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #UP
  } \lyricmode { up }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #CENTER
  } \lyricmode { center }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #DOWN
  } \lyricmode { down }
  \new Staff { c'1 }
>>
```



Vedi anche

Installed Files: ‘ly/engraver-init.ly’, ‘scm/define-grobs.scm’.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

Internals Reference: Sezione “Contexts” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*.

4.4.2 Explicit staff and system positioning

One way to understand the flexible vertical spacing mechanisms explained above is as a collection of settings that control the amount of vertical padding between staves and systems.

It is possible to approach vertical spacing in a different way using `NonMusicalPaperColumn.line-break-system-details`. While the flexible vertical spacing mechanisms specify vertical padding, `NonMusicalPaperColumn.line-break-system-details` can specify exact vertical positions on the page.

`NonMusicalPaperColumn.line-break-system-details` accepts an associative list of three different settings:

- `X-offset`
- `Y-offset`
- `alignment-distances`

Grob overrides, including the overrides for `NonMusicalPaperColumn` below, can occur in any of three different places in an input file:

- in the middle of note entry directly
- in a `\context` block
- in the `\with` block

When we override `NonMusicalPaperColumn`, we use the usual `\override` command in `\context` blocks and in the `\with` block. On the other hand, when we override `NonMusicalPaperColumn` in the middle of note entry, use the special `\overrideProperty` command. Here are some example `NonMusicalPaperColumn` overrides with the special `\overrideProperty` command:

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 40))
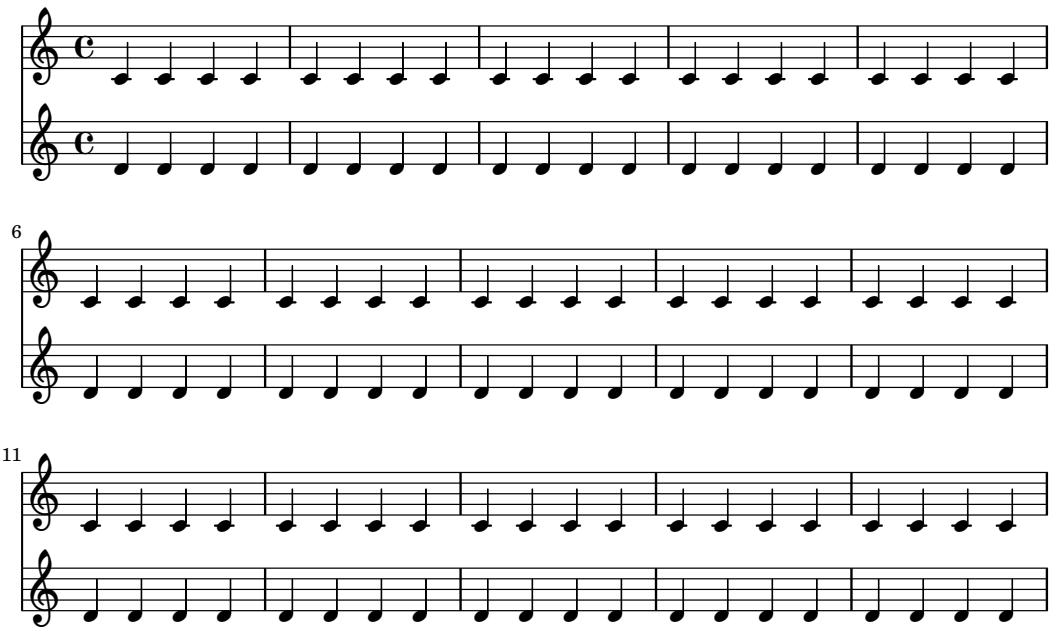
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
     (Y-offset . 40))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((alignment-distances . (15)))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
     (Y-offset . 40)
     (alignment-distances . (15)))
```

To understand how each of these different settings work, we begin by looking at an example that includes no overrides at all.
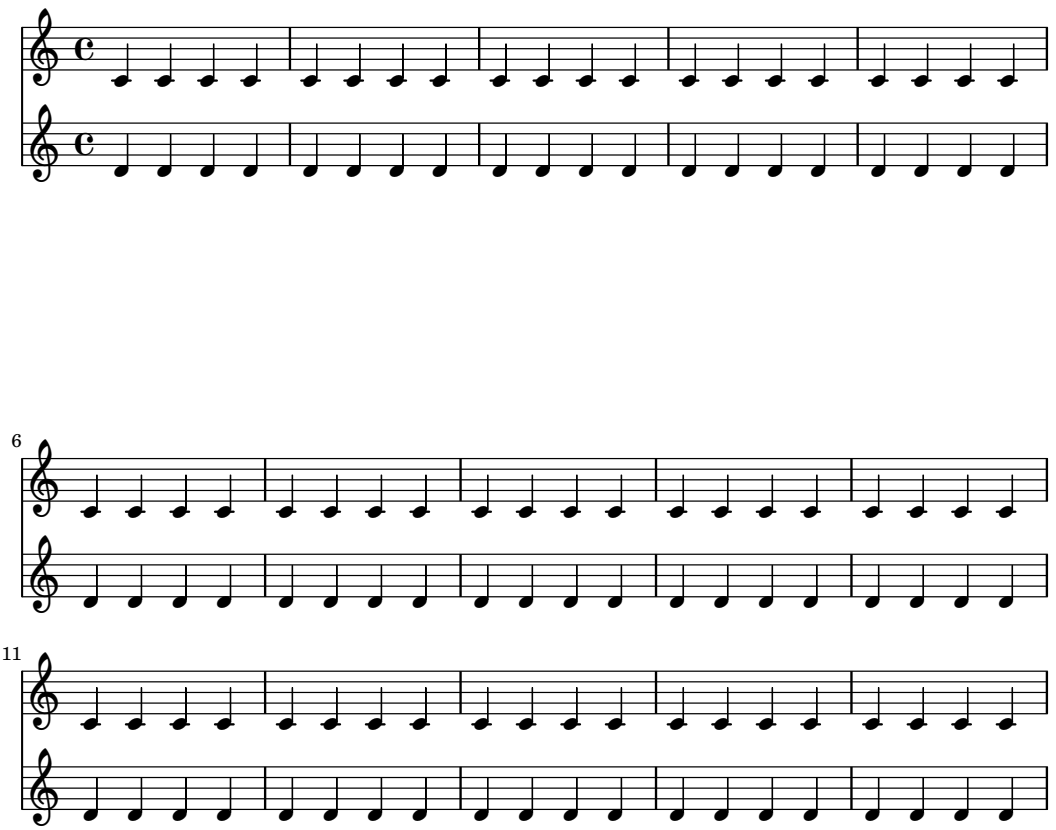
```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```



This score isolates line- and page-breaking information in a dedicated voice. This technique of creating a breaks voice will help keep layout separate from music entry as our example becomes more complicated. See Sezione 4.3.8 [Using an extra voice for breaks], pagina 521.

Explicit `\breaks` evenly divide the music into six measures per line. Vertical spacing results from LilyPond's defaults. To set the vertical startpoint of each system explicitly, we can set the `Y-offset` pair in the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob:

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 0))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 40))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 60))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```

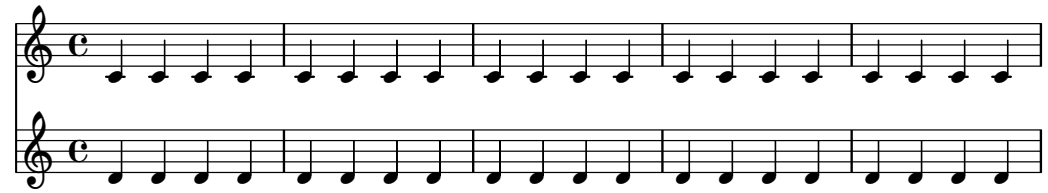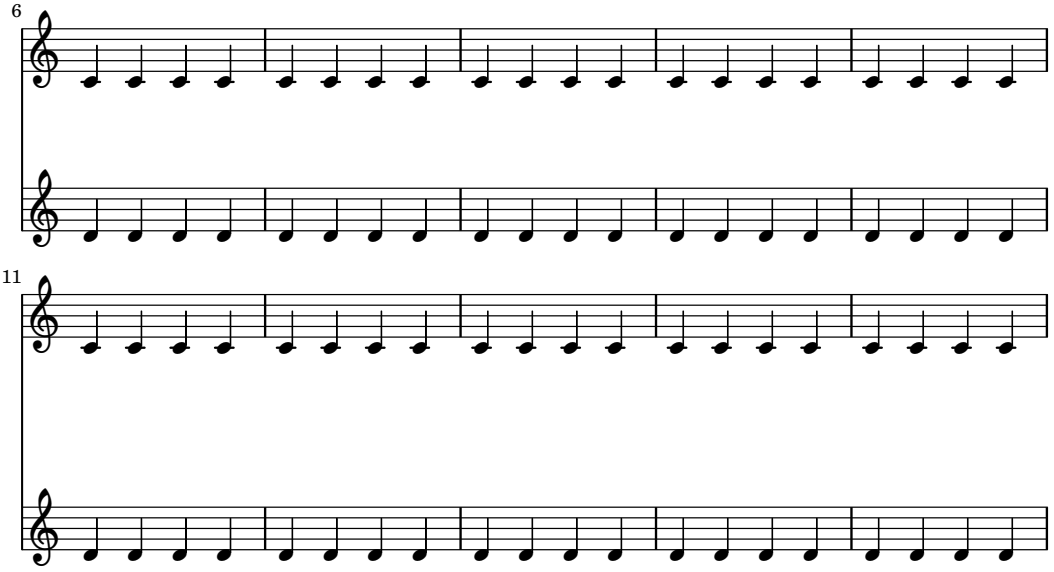


Note that `line-break-system-details` takes an associative list of potentially many values, but that we set only one value here. Note, too, that the `Y-offset` property here determines the exact vertical position on the page at which each new system will render.

Now that we have set the vertical startpoint of each system explicitly, we can also set the vertical distances between staves within each system manually. We do this using the `alignment-distances` subproperty of `line-break-system-details`.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 20)
               (alignment-distances . (10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 60)
               (alignment-distances . (15)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 85)
             (alignment-distances . (20)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
```
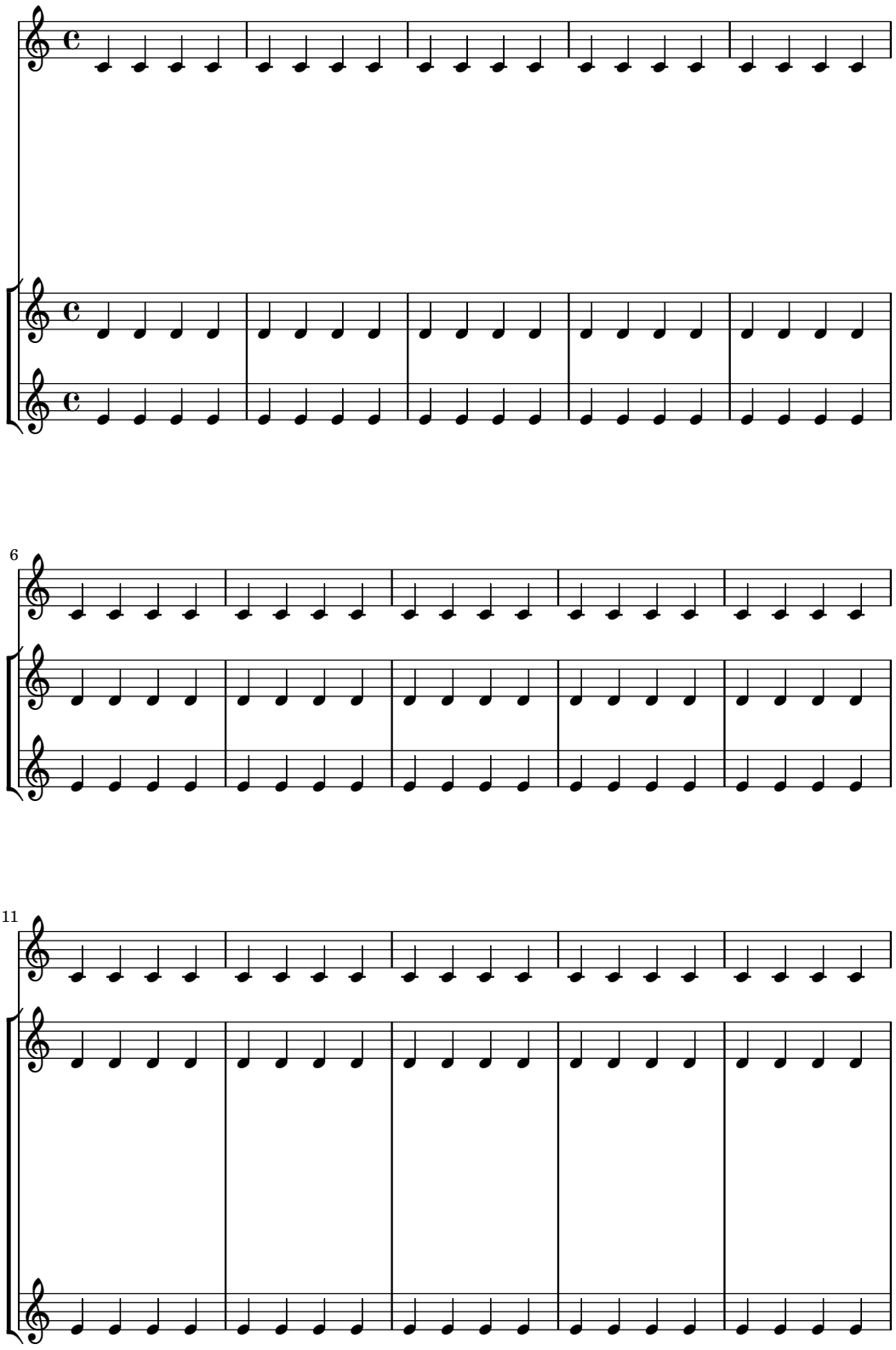
```
      }
    >>
  }
}
```





Note that here we assign two different values to the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob. Though the `line-break-system-details` attribute alist accepts many additional spacing parameters (including, for example, a corresponding `X-offset` pair), we need only set the `Y-offset` and `alignment-distances` pairs to control the vertical startpoint of every system and every staff. Finally, note that `alignment-distances` specifies the vertical positioning of staves but not of staff groups.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
```

```
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 0)
               (alignment-distances . (30 10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 60)
               (alignment-distances . (10 10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 100)
               (alignment-distances . (10 30)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new StaffGroup <<
        \new Staff { \repeat unfold 15 { d'4 d' d' d' } }
        \new Staff { \repeat unfold 15 { e'4 e' e' e' } }
      >>
    >>
  }
}
```



Some points to consider:

- When using `alignment-distances`, lyrics and other non-staff lines do not count as a staff.
- The units of the numbers passed to `X-offset`, `Y-offset` and `alignment-distances` are interpreted as multiples of the distance between adjacent staff lines. Positive values move staves and lyrics up, negative values move staves and lyrics down.
- Because the `NonMusicalPaperColumn.line-break-system-details` settings given here allow the positioning of staves and systems anywhere on the page, it is possible to violate

paper or margin boundaries or even to print staves or systems on top of one another. Reasonable values passed to these different settings will avoid this.

Vedi anche

Snippets: Sezione "Spacing" in *Frammenti di codice*.

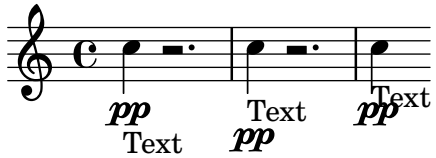
4.4.3 Vertical collision avoidance

Intuitively, there are some objects in musical notation that belong to the staff and there are other objects that should be placed outside the staff. Objects belonging outside the staff include things such as rehearsal marks, text and dynamic markings (from now on, these will be called outside-staff objects). LilyPond's rule for the vertical placement of outside-staff objects is to place them as close to the staff as possible but not so close that they collide with another object.

LilyPond uses the `outside-staff-priority` property to determine whether a grob is an outside-staff object: if `outside-staff-priority` is a number, the grob is an outside-staff object. In addition, `outside-staff-priority` tells LilyPond in which order the objects should be placed.

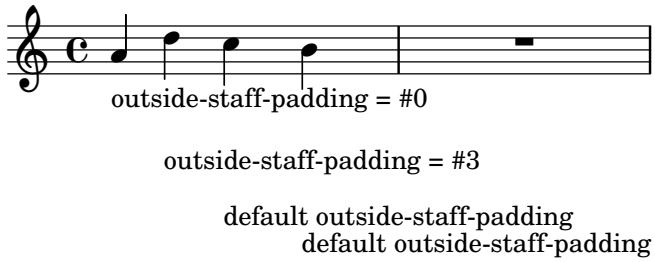
First, LilyPond places all the objects that do not belong outside the staff. Then it sorts the outside-staff objects according to their `outside-staff-priority` (in increasing order). One by one, LilyPond takes the outside-staff objects and places them so that they do not collide with any objects that have already been placed. That is, if two outside-staff grobs are competing for the same space, the one with the lower `outside-staff-priority` will be placed closer to the staff.

```
c4_"Text"\pp
r2.
\once \override TextScript.outside-staff-priority = #1
c4_"Text"\pp % this time the text will be closer to the staff
r2.
% by setting outside-staff-priority to a non-number,
% we disable the automatic collision avoidance
\once \override TextScript.outside-staff-priority = ##f
\once \override DynamicLineSpanner.outside-staff-priority = ##f
c4_"Text"\pp % now they will collide
```



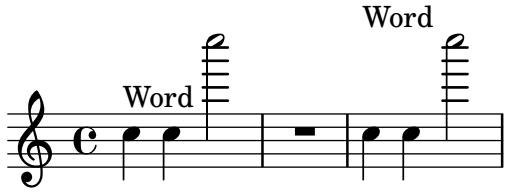
The vertical padding around outside-staff objects can be controlled with `outside-staff-padding`.

```
\once \override TextScript.outside-staff-padding = #0
a4-"outside-staff-padding = #0"
\once \override TextScript.outside-staff-padding = #3
d-"outside-staff-padding = #3"
c-"default outside-staff-padding"
b-"default outside-staff-padding"
R1
```



By default, outside-staff objects are placed so they avoid a horizontal collision with previously-positioned grobs. This can lead to situations in which objects are placed close to each other horizontally. As shown in the example below, setting `outside-staff-horizontal-padding` increases the horizontal spacing required, and in this case moves the text up to prevent it from getting too close to the ledger lines.

```
c4^"Word" c c''2
R1
\once \override TextScript.outside-staff-horizontal-padding = #1
c,,4^"Word" c c''2
```



Vedi anche

Snippets: Sezione “Spacing” in *Frammenti di codice*.
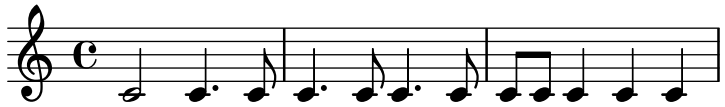
4.5 Horizontal spacing

4.5.1 Horizontal spacing overview

The spacing engine translates differences in durations into stretchable distances (‘springs’) of differing lengths. Longer durations get more space, shorter durations get less. The shortest durations get a fixed amount of space (which is controlled by `shortest-duration-space` in the Sezione “SpacingSpanner” in *Guida al Funzionamento Interno* object). The longer the duration, the more space it gets: doubling a duration adds a fixed amount (this amount is controlled by `spacing-increment`) of space to the note.

For example, the following piece contains lots of half, quarter, and 8th notes; the eighth note is followed by 1 note head width (NHW). The quarter note is followed by 2 NHW, the half by 3 NHW, etc.

```
c2 c4. c8
c4. c8 c4. c8
c8 c c4 c c
```



Normally, `spacing-increment` is set to 1.2 staff space, which is approximately the width of a note head, and `shortest-duration-space` is set to 2.0, meaning that the shortest note gets 2.4 staff space (2.0 times the `spacing-increment`) of horizontal space. This space is counted from the left edge of the symbol, so the shortest notes are generally followed by one NHW of space.
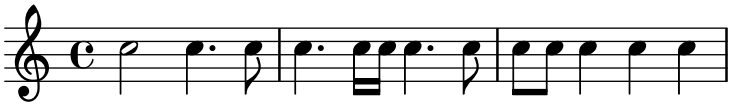
If one would follow the above procedure exactly, then adding a single 32nd note to a score that uses 8th and 16th notes, would widen up the entire score a lot. The shortest note is no longer a 16th, but a 32nd, thus adding 1 NHW to every note. To prevent this, the shortest duration for spacing is not the shortest note in the score, but rather the one which occurs most frequently.

The most common shortest duration is determined as follows: in every measure, the shortest duration is determined. The most common shortest duration is taken as the basis for the spacing, with the stipulation that this shortest duration should always be equal to or shorter than an 8th note. The shortest duration is printed when you run `lilypond` with the '`--verbose`' option.
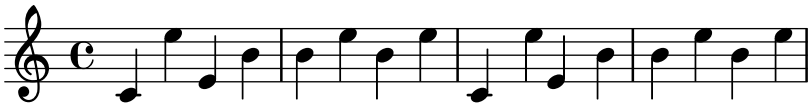
These durations may also be customized. If you set the `common-shortest-duration` in Sezione "SpacingSpanner" in *Guida al Funzionamento Interno*, then this sets the base duration for spacing. The maximum duration for this base (normally an 8th), is set through `base-shortest-duration`.

Notes that are even shorter than the common shortest note are followed by a space that is proportional to their duration relative to the common shortest note. So if we were to add only a few 16th notes to the example above, they would be followed by half a NHW:

```
c2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c
```



In the *Essay on automated music engraving*, it was explained that stem directions influence spacing (see Sezione "Optical spacing" in *Saggio*). This is controlled with the `stem-spacing-correction` property in the Sezione "NoteSpacing" in *Guida al Funzionamento Interno*, object. These are generated for every Sezione "Voice" in *Guida al Funzionamento Interno* context. The `StaffSpacing` object (generated in Sezione "Staff" in *Guida al Funzionamento Interno* context) contains the same property for controlling the stem/bar line spacing. The following example shows these corrections, once with default settings, and once with exaggerated corrections:



Proportional notation is supported; see Sezione 4.5.5 [Proportional notation], pagina 543.

Vedi anche

Essay on automated music engraving: Sezione "Optical spacing" in *Saggio*.

Snippets: Sezione "Spacing" in *Frammenti di codice*.

Internals Reference: Sezione "SpacingSpanner" in *Guida al Funzionamento Interno*, Sezione "NoteSpacing" in *Guida al Funzionamento Interno*, Sezione "StaffSpacing" in *Guida al Funzionamento Interno*, Sezione "NonMusicalPaperColumn" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

There is no convenient mechanism to manually override spacing. The following work-around may be used to insert extra space into a score, adjusting the padding value as necessary.

```
\override Score.NonMusicalPaperColumn.padding = #10
```
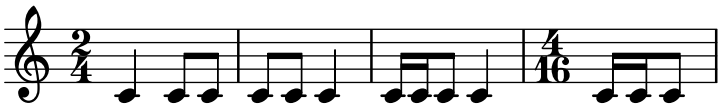
No work-around exists for decreasing the amount of space.

4.5.2 New spacing area

New sections with different spacing parameters can be started with `newSpacingSection`. This is useful when there are sections with a different notions of long and short notes.
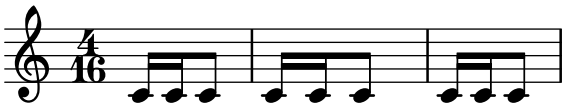
In the following example, the time signature change introduces a new section, and hence the 16ths notes are automatically spaced slightly wider.

```
\time 2/4
c4 c8 c
c8 c c4 c16[ c c8] c4
\newSpacingSection
\time 4/16
c16[ c c8]
```



The `\newSpacingSection` command creates a new `SpacingSpanner` object at that musical moment. If the automatic spacing adjustments do not give the required spacing, manual `\overrides` may be applied to its properties. These must be applied at the same musical moment as the `\newSpacingSection` command itself. They will then affect the spacing of all the following music until the properties are changed in a new spacing section, for example,

```
\time 4/16
c16[ c c8]
\newSpacingSection
\override Score.SpacingSpanner.spacing-increment = #2
c16[ c c8]
\newSpacingSection
\revert Score.SpacingSpanner.spacing-increment
c16[ c c8]
```



Vedi anche

Snippets: Sezione “Spacing” in *Frammenti di codice*.
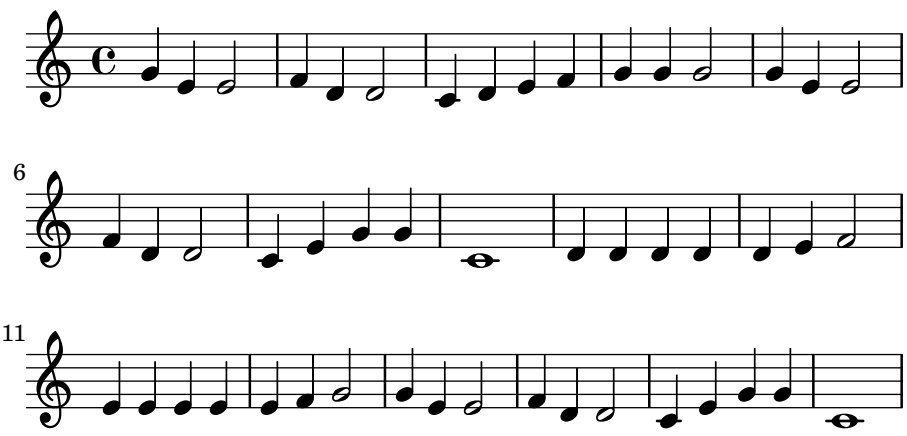
Internals Reference: Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*.

4.5.3 Changing horizontal spacing
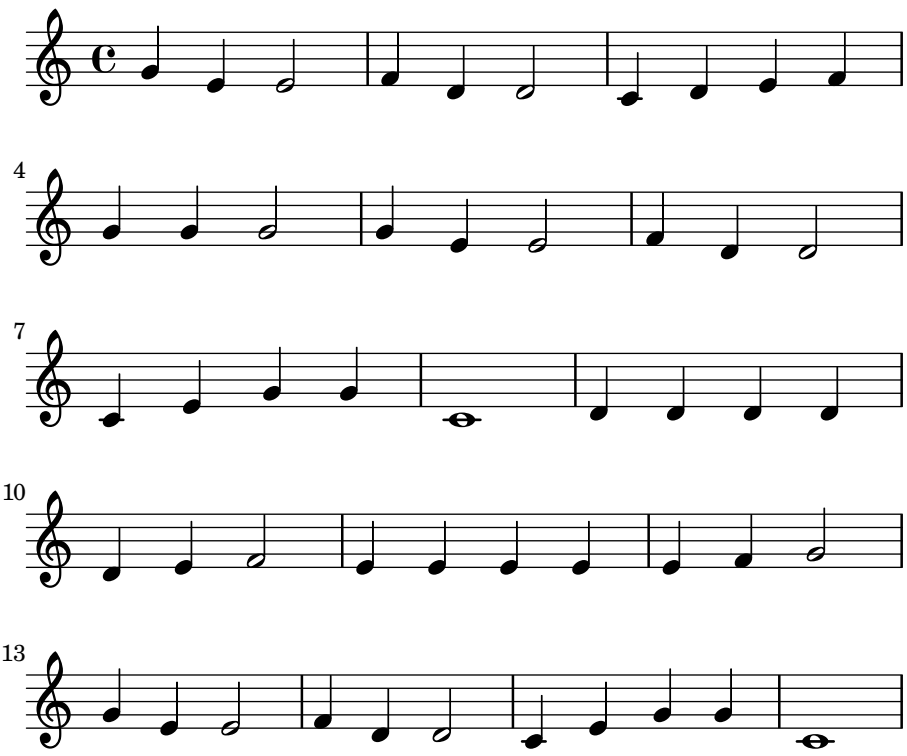
Horizontal spacing may be altered with the `base-shortest-duration` property. Here we compare the same music; once without altering the property, and then altered. Larger values of `ly:make-moment` will produce smaller music. Note that `ly:make-moment` constructs a duration, so `1 4` is a longer duration than `1 16`.

```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
```

```
}
```



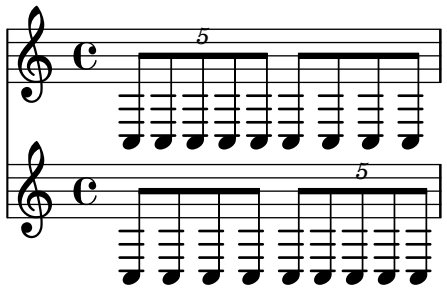
```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
    }
  }
}
```



Frammenti di codice selezionati

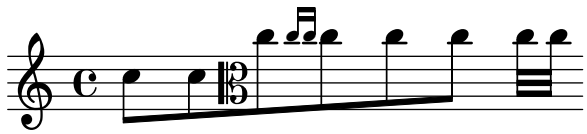
By default, spacing in tuplets depends on various non-duration factors (such as accidentals, clef changes, etc). To disregard such symbols and force uniform equal-duration spacing, use `Score.SpacingSpanner.uniform-stretching`. This property can only be changed at the beginning of a score,

```
\score {
  <<
    \new Staff {
      \tuplet 5/4 { c8 c c c c } c8 c c c
    }
    \new Staff {
      c8 c c c \tuplet 5/4 { c8 c c c c }
    }
  >>
  \layout {
    \context {
      \Score
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```



When `strict-note-spacing` is set, notes are spaced without regard for clefs, bar lines, and grace notes,

```
\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff { c8[ c \clef alto c \grace { c16 c } c8 c c]  c32[ c] }
```



Vedi anche

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.5.4 Line length

The most basic settings influencing the spacing are `indent` and `line-width`. They are set in the `\layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `\layout` block, then systems ends at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for

short fragments, and for checking how tight the natural spacing is. The normal default setting is false, but if the score has only one system the default value is true.

The option `ragged-last` is similar to `ragged-right`, but only affects the last line of the piece. No restrictions are put on that line. The result is similar to formatting text paragraphs. In a paragraph, the last line simply takes its natural horizontal length.

```
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
```

Vedi anche

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.5.5 Proportional notation

LilyPond supports proportional notation, a type of horizontal spacing in which each note consumes an amount of horizontal space exactly equivalent to its rhythmic duration. This type of proportional spacing is comparable to horizontal spacing on top of graph paper. Some late 20th- and early 21st-century scores use proportional notation to clarify complex rhythmic relationships or to facilitate the placement of timelines or other graphics directly in the score.
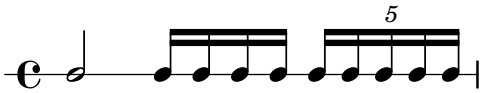
LilyPond supports five different settings for proportional notation, which may be used together or alone:

- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove "Separating_line_group_engraver"`
- `\override PaperColumn.used = ##t`

In the examples that follow, we explore these five different proportional notation settings and examine how these settings interact.

We start with the following one-measure example, which uses classical spacing with ragged-right turned on.

```
\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
  >>
}
```



Notice that the half note which begins the measure takes up far less than half of the horizontal space of the measure. Likewise, the sixteenth notes and sixteenth-note quintuplets (or twentieth notes) which end the measure together take up far more than half the horizontal space of the measure.

In classical engraving, this spacing may be exactly what we want because we can borrow horizontal space from the half note and conserve horizontal space across the measure as a whole.

On the other hand, if we want to insert a measured timeline or other graphic above or below our score, we need proportional notation. We turn proportional notation on with the proportionalNotationDuration setting.

```
\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
  >>
 \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```



The half note at the beginning of the measure and the faster notes in the second half of the measure now occupy equal amounts of horizontal space. We could place a measured timeline or graphic above or below this example.

The `proportionalNotationDuration` setting is a context setting that lives in `Score`. Remember that context settings can appear in one of three locations within our input file – in a `\with` block, in a `\context` block, or directly in music entry preceded by the `\set` command. As with all context settings, users can pick which of the three different locations they would like to set `proportionalNotationDuration` in to.

The `proportionalNotationDuration` setting takes a single argument, which is the reference duration against that all music will be spaced. The LilyPond Scheme function `make-moment` takes two arguments – a numerator and denominator which together express some fraction of a whole note. The call `(ly:make-moment 1/20)` therefore produces a reference duration of a twentieth note. Values such as `(ly:make-moment 1/16)`, `(ly:make-moment 1/8)`, and `(ly:make-moment 3/97)` are all possible as well.

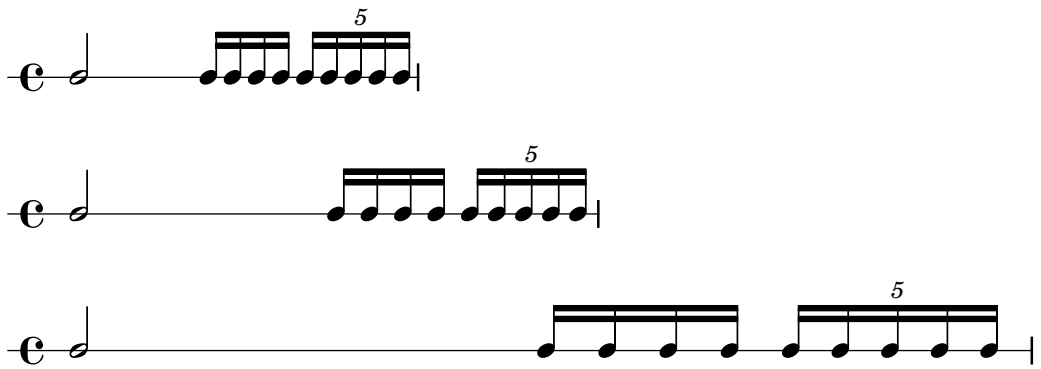
How do we select the right reference duration to pass to `proportionalNotationDuration`? Usually by a process of trial and error, beginning with a duration close to the fastest (or smallest) duration in the piece. Smaller reference durations space music loosely; larger reference durations space music tightly.

```
\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/8)
    }
  }
```

```
}

\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/16)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/32)
    }
  }
}
```



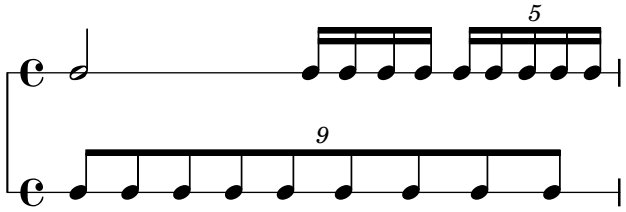
Note that too large a reference duration – such as the eighth note, above – spaces music too tightly and can cause note head collisions. Also that proportional notation in general takes up more horizontal space than classical spacing. Proportional spacing provides rhythmic clarity at the expense of horizontal space.

Next we examine how to optimally space overlapping tuplets.

We start by examining what happens to our original example, with classical spacing, when we add a second staff with a different type of tuplet.

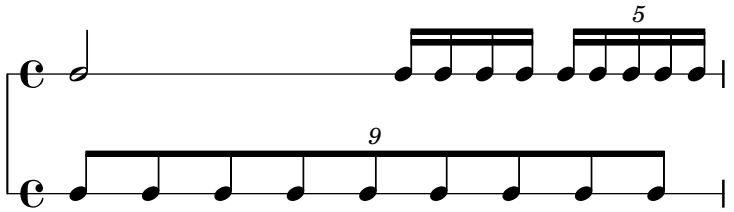
```
\score {
  <<
    \new RhythmicStaff {
```

```
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c'8 c' c' c' c' c' c' c' c' }
    }
  >>
}
```



The spacing is bad because the evenly spaced notes of the bottom staff do not stretch uniformly. Classical engravings include very few complex triplets and so classical engraving rules can generate this type of result. Setting `proportionalNotationDuration` fixes this.

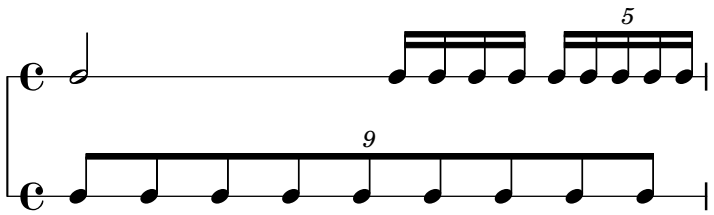
```
\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c'8 c' c' c' c' c' c' c' c' }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```



But if we look very carefully we can see that notes of the second half of the 9-tuplet space ever so slightly more widely than the notes of the first half of the 9-tuplet. To ensure uniform stretching, we turn on `uniform-stretching`, which is a property of `SpacingSpanner`.

```
\score {
  <<
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c'8 c' c' c' c' c' c' c' c' }
```

```
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```



Our two-staff example now spaces exactly, our rhythmic relationships are visually clear, and we can include a measured timeline or graphic if we want.

Note that the LilyPond's proportional notation package expects that all proportional scores set the SpacingSpanner's 'uniform-stretching attribute to ##t. Setting proportionalNotation-Duration without also setting the SpacingSpanner's 'uniform-stretching attribute to ##t will, for example, cause Skips to consume an incorrect amount of horizontal space.

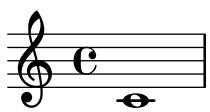
The SpacingSpanner is an abstract grob that lives in the Score context. As with our settings of proportionalNotationDuration, overrides to the SpacingSpanner can occur in any of three different places in our input file – in the Score \with block, in a Score \context block, or in note entry directly.
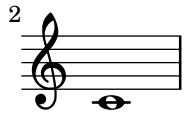
There is by default only one `SpacingSpanner` per `Score`. This means that, by default, `uniform-stretching` is either turned on for the entire score or turned off for the entire score. We can, however, override this behavior and turn on different spacing features at different places in the score. We do this with the command `\newSpacingSection`. See Sezione 4.5.2 [New spacing area], pagina 540, for more info.

Next we examine the effects of the `Separating_line_group_engraver` and see why proportional scores frequently remove this engraver. The following example shows that there is a small amount of "prefatory" space just before the first note in each system.

```
\paper {
  indent = #0
}

\new Staff {
  c'1
  \break
  c'1
}
```

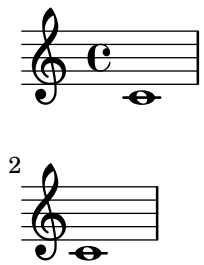




The amount of this prefatory space is the same whether after a time signature, a key signature or a clef. `Separating_line_group_engraver` is responsible for this space. Removing `Separating_line_group_engraver` reduces this space to zero.

```
\paper {
  indent = #0
}

\new Staff \with {
  \remove "Separating_line_group_engraver"
} {
  c'1
  \break
  c'1
}
```
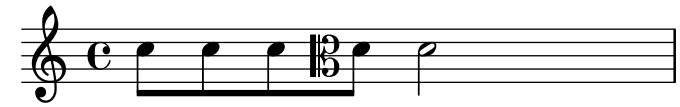


non-musical elements like time signatures, key signatures, clefs and accidentals are problematic in proportional notation. None of these elements has rhythmic duration. But all of these elements consume horizontal space. Different proportional scores approach these problems differently.
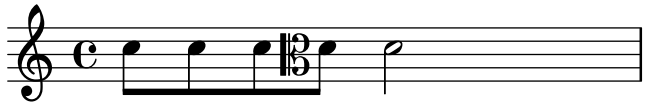
It may be possible to avoid spacing problems with key signatures simply by not having any. This is a valid option since most proportional scores are contemporary music. The same may be true of time signatures, especially for those scores that include a measured timeline or other graphic. But these scores are exceptional and most proportional scores include at least some time signatures. Clefs and accidentals are even more essential.

So what strategies exist for spacing non-musical elements in a proportional context? One good option is the `strict-note-spacing` property of `SpacingSpanner`. Compare the two scores below:

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  c''8 c'' c'' \clef alto d' d'2
}

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  c''8 c'' c'' \clef alto d' d'2
}
```





Both scores are proportional, but the spacing in the first score is too loose because of the clef change. The spacing of the second score remains strict, however, because strict-note-spacing is turned on. Turning on strict-note-spacing causes the width of time signatures, key signatures, clefs and accidentals to play no part in the spacing algorithm.

In addition to the settings given here, there are other settings that frequently appear in proportional scores. These include:

- `\override SpacingSpanner.strict-grace-spacing = ##t`
- `\set tupletFullLength = ##t`
- `\override Beam.breakable = ##t`
- `\override Glissando.breakable = ##t`
- `\override TextSpanner.breakable = ##t`
- `\remove "Forbid_line_break_engraver" in the Voice context`

These settings space grace notes strictly, extend tuplet brackets to mark both rhythmic start- and stop-points, and allow spanning elements to break across systems and pages. See the respective parts of the manual for these related settings.

Vedi anche

Notation Reference: Sezione 4.5.2 [New spacing area], pagina 540.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.6 Fitting music onto fewer pages

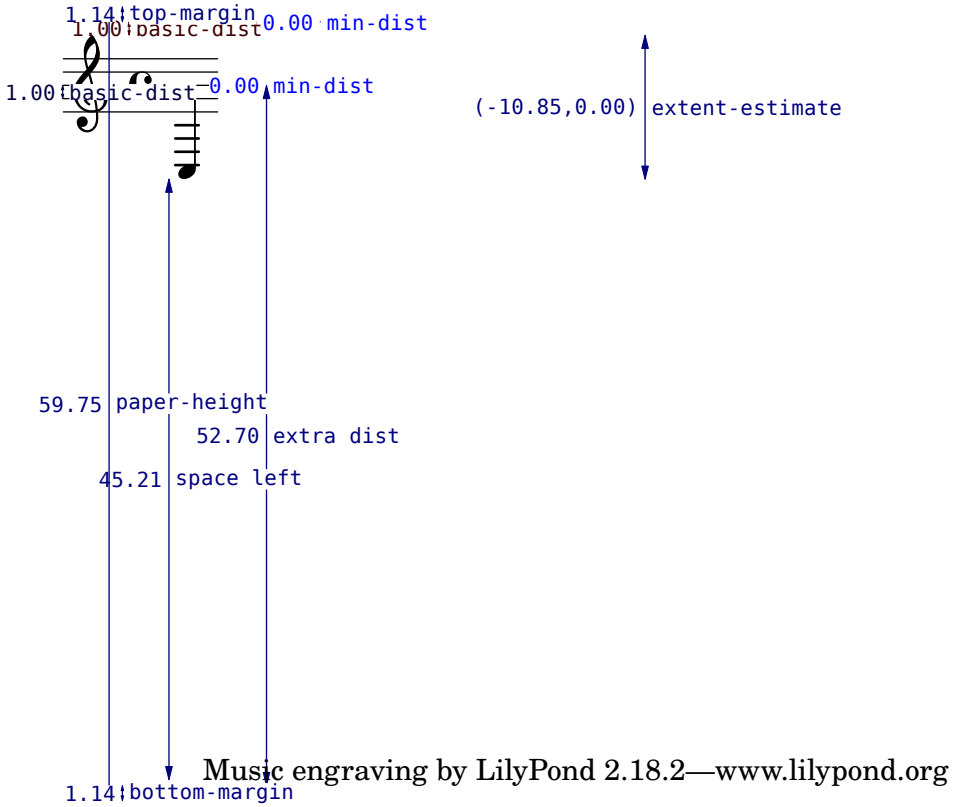
Sometimes you can end up with one or two staves on a second (or third, or fourth. . .) page. This is annoying, especially if you look at previous pages and it looks like there is plenty of room left on those.

When investigating layout issues, `annotate-spacing` is an invaluable tool. This command prints the values of various layout spacing variables; for more details see the following section, Sezione 4.6.1 [Displaying spacing], pagina 549.

4.6.1 Displaying spacing

To graphically display the dimensions of vertical layout variables that may be altered for page formatting, set `annotate-spacing` in the `\paper` block:

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



All layout dimensions are displayed in staff-spaces, regardless of the units specified in the `\paper` or `\layout` block. In the above example, `paper-height` has a value of 59.75 `staff-spaces`, and the `staff-size` is 20 points (the default value). Note that:

1 point = (25.4/72.27) mm

1 staff-space = (`staff-size`)/4 pts
= (`staff-size`)/4 * (25.4/72.27) mm

In this case, one `staff-space` is approximately equal to 1.757mm. Thus the `paper-height` measurement of 59.75 `staff-spaces` is equivalent to 105 millimeters, the height of `a6` paper in landscape orientation. The pairs (*a*,*b*) are intervals, where *a* is the lower edge and *b* the upper edge of the interval.

Vedi anche

Notation Reference: Sezione 4.2.2 [Setting the staff size], pagina 514.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

4.6.2 Changing spacing

The output of `annotate-spacing` reveals vertical dimensions in great detail. For details about modifying margins and other layout variables, see Sezione 4.1 [Page layout], pagina 502.

Other than margins, there are a few other options to save space:

- Force systems to move as close together as possible (to fit as many systems as possible onto a page) while being spaced so that there is no blank space at the bottom of the page.

```
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Force the number of systems. This can help in two ways. Just setting a value, even the same value as the number of systems being typeset by default, will sometimes cause more systems to be fitted onto each page, as an estimation step is then bypassed, giving a more accurate fit to each page. Also, forcing an actual reduction in the number of systems may save a further page. For example, if the default layout has 11 systems, the following assignment will force a layout with 10 systems.
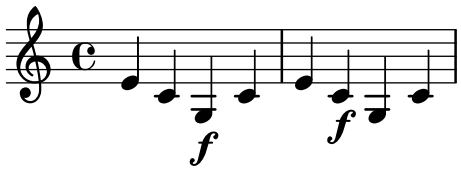
```
\paper {
  system-count = #10
}
```

- Force the number of pages. For example, the following assignment will force a layout with 2 pages.

```
\paper {
  page-count = #2
}
```
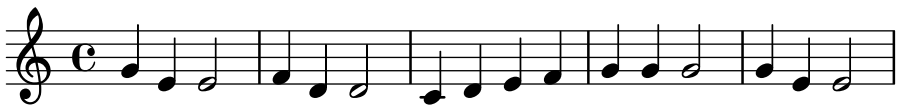
- Avoid (or reduce) objects that increase the vertical size of a system. For example, volta repeats (or alternate repeats) require extra space. If these repeats are spread over two systems, they will take up more space than one system with the volta repeats and another system without. For example, dynamics that 'stick out' of a system can be moved closer to the staff:

```
e4 c g\f c
e4 c g-\tweak X-offset #-2.7 \f c
```



- Alter the horizontal spacing via `SpacingSpanner`. For more details, see Sezione 4.5.3 [Changing horizontal spacing], pagina 540. The following example illustrates the default spacing:
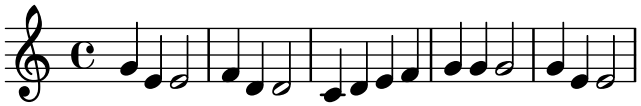
```
\score {
  \relative c'' {
    g4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
}
```



The next example modifies `common-shortest-duration` from a value of `1/4` to `1/2`. The quarter note is the most common and shortest duration in this example, so by making this duration longer, a 'squeezing' effect occurs:

```
\score {
  \relative c'' {
    g4 e e2 |
    f4 d d2 |
```

```
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.common-shortest-duration =
        #(ly:make-moment 1/2)
    }
  }
}
```



The `common-shortest-duration` property cannot be modified dynamically, so it must always be placed in a `\context` block so that it applies to the whole score.

Vedi anche

Notation Reference: Sezione 4.1 [Page layout], pagina 502, Sezione 4.5.3 [Changing horizontal spacing], pagina 540.

Snippets: Sezione “Spacing” in *Frammenti di codice*.

5 Changing defaults

The purpose of LilyPond's design is to provide the finest quality output by default. Nevertheless, it may happen that you need to change this default layout. The layout is controlled through a large number of 'knobs and switches' collectively called 'properties'. A tutorial introduction to accessing and modifying these properties can be found in the Learning Manual, see Sezione "Tweaking output" in *Manuale di Apprendimento*. This should be read first. This chapter covers similar ground, but in a style more appropriate to a reference manual.

The definitive description of the controls available for tuning can be found in a separate document: Sezione "the Internals Reference" in *Guida al Funzionamento Interno*. That manual lists all the variables, functions and options available in LilyPond. It is written as a HTML document, which is available on-line, and is also included with the LilyPond documentation package.

Internally, LilyPond uses Scheme (a LISP dialect) to provide infrastructure. Overriding layout decisions in effect accesses the program internals, which requires Scheme input. Scheme elements are introduced in a '`.ly`' file with the hash mark `#`.[1]

5.1 Interpretation contexts

This section describes what contexts are, and how to modify them.

Vedi anche

Learning Manual: Sezione "Contexts and engravers" in *Manuale di Apprendimento*.

Installed Files: '`ly/engraver-init.ly`', '`ly/performer-init.ly`'.

Snippets: Sezione "Contexts and engravers" in *Frammenti di codice*.

Internals Reference: Sezione "Contexts" in *Guida al Funzionamento Interno*, Sezione "Engravers and Performers" in *Guida al Funzionamento Interno*.

5.1.1 Contexts explained

Contexts are arranged hierarchically:

Output definitions - blueprints for contexts

This section explains the relevance of output definitions when working with contexts. Examples for actual output definitions are given later (see [Changing all contexts of the same type], pagina 563).

While music written in a file may refer to context types and names, contexts are created only when the music is actually being interpreted. LilyPond interprets music under control of an 'output definition' and may do so for several different output definitions, resulting in different output. The output definition relevant for printing music is specified using `\layout`.

A much simpler output definition used for producing Midi output is specified using `\midi`. Several other output definitions are used by LilyPond internally, like when using the part combiner (⟨undefined⟩ [Automatic part combining], pagina ⟨undefined⟩) or creating music quotes (⟨undefined⟩ [Quoting other voices], pagina ⟨undefined⟩).

Output definitions define the relation between contexts as well as their respective default settings. While most changes will usually be made inside of a `\layout` block, Midi-related settings will only have an effect when made within a `\midi` block.

Some settings affect several outputs: for example, if `autoBeaming` is turned off in some context, beams count as melismata for the purpose of matching music to lyrics as described in

[1] Sezione "Scheme tutorial" in *Estendere*, contains a short tutorial on entering numbers, lists, strings, and symbols in Scheme.

[Automatic syllable durations], pagina 251. This matching is done both for printed output as well as for Midi. If changes made to `autoBeaming` within a context definition of a `\layout` block are not repeated in the corresponding `\midi` block, lyrics and music will get out of sync in Midi.

Vedi anche

Installed Files: '`ly/engraver-init.ly`'. '`ly/performer-init.ly`'.

Score - the master of all contexts

This is the top level notation context. No other context can contain a Score context. By default the Score context handles the administration of time signatures and makes sure that items such as clefs, time signatures, and key-signatures are aligned across staves.

A Score context is instantiated implicitly when a `\score {...}` block is processed.

Top-level contexts - staff containers

StaffGroup

Groups staves while adding a bracket on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically. `StaffGroup` only consists of a collection of staves, with a bracket in front and spanning bar lines.

ChoirStaff

Identical to `StaffGroup` except that the bar lines of the contained staves are not connected vertically.

GrandStaff

A group of staves, with a brace on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically.

PianoStaff

Just like `GrandStaff`, but with support for instrument names to the left of each system.

Intermediate-level contexts - staves

Staff

Handles clefs, bar lines, keys, accidentals. It can contain `Voice` contexts.

RhythmicStaff

Like `Staff` but for printing rhythms. Pitches are ignored when engraving; the notes are printed on one line. The MIDI rendition retains pitches unchanged.

TabStaff

Context for generating tablature. By default lays the music expression out as a guitar tablature, printed on six lines.

DrumStaff

Handles typesetting for percussion. Can contain `DrumVoice`

VaticanaStaff

Same as `Staff`, except that it is designed for typesetting a piece in gregorian style.

MensuralStaff

Same as `Staff`, except that it is designed for typesetting a piece in mensural style.

Bottom-level contexts - voices

Voice-level contexts initialise certain properties and start appropriate engravers. A bottom-level context is one without `defaultchild`. While it is possible to let it accept/contain subcontexts, they can only be created and entered explicitly.

Voice

Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and sub-scripts, slurs, ties, and rests. You have to instantiate this explicitly if you require multiple voices on the same staff.

VaticanaVoice

Same as `Voice`, except that it is designed for typesetting a piece in gregorian style.

MensuralVoice

Same as `Voice`, with modifications for typesetting a piece in mensural style.

Lyrics

Corresponds to a voice with lyrics. Handles the printing of a single line of lyrics.

DrumVoice

The voice context used in a percussion staff.

FiguredBass

The context in which `BassFigure` objects are created from input entered in `\figuremode` mode.

TabVoice

The voice context used within a `TabStaff` context. Usually left to be created implicitly.

CueVoice

A voice context used to render notes of a reduced size, intended primarily for adding cue notes to a staff, see ⟨undefined⟩ [Formatting cue notes], pagina ⟨undefined⟩. Usually left to be created implicitly.

ChordNames

Typesets chord names.

5.1.2 Creating and referencing contexts

LilyPond will create lower-level contexts automatically if a music expression is encountered before a suitable context exists, but this is usually successful only for simple scores or music fragments like the ones in the documentation. For more complex scores it is advisable to specify all contexts explicitly with either the `\new` or `\context` command. The syntax of these two commands is very similar:
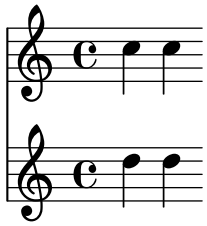
`[\new | \context]` *`Context`* `[ =` *`name`*`] [`*`music-expression`*`]`

where either `\new` or `\context` may be specified. *Context* is the type of context which is to be created, *name* is an optional name to be given to the particular context being created and *music-expression* is a single music expression that is to be interpreted by the engravers and performers in this context.

The `\new` prefix without a name is commonly used to create scores with many staves:
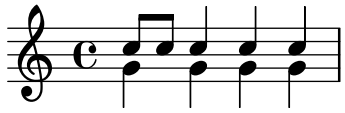
```
<<
  \new Staff {
    % leave the Voice context to be created implicitly
    c4 c
  }
  \new Staff {
```

```
    d4 d
  }
>>
```



and to place several voices into one staff:

```
<<
  \new Staff <<
    \new Voice {
      \voiceOne
      c8 c c4 c c
    }
    \new Voice {
      \voiceTwo
      g4 g g g
    }
  >>
>>
```
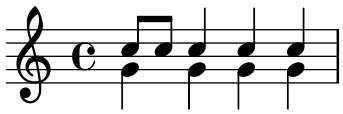


`\new` should always be used to specify unnamed contexts.

The difference between `\new` and `\context` is in the action taken:

- `\new` with or without a name will always create a fresh, distinct, context, even if one with the same name already exists:

```
<<
  \new Staff <<
    \new Voice = "A" {
      \voiceOne
      c8 c c4 c c
    }
    \new Voice = "A" {
      \voiceTwo
      g4 g g g
    }
  >>
>>
```
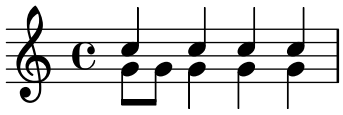


- `\context` with a name specified will create a distinct context only if a context of the same type with the same name in the same context hierarchy does not already exist. Otherwise

it will be taken as a reference to that previously created context, and its music expression will be passed to that context for interpretation.

One application of named contexts is in separating the score layout from the musical content. Either of these two forms is valid:

```
\score {
  <<
    % score layout
    \new Staff <<
      \new Voice = "one" {
        \voiceOne
      }
      \new Voice = "two" {
        \voiceTwo
      }
    >>
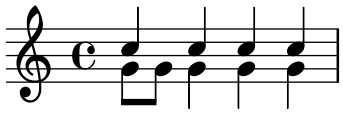
    % musical content
    \context Voice = "one" {
      \relative c'' {
        c4 c c c
      }
    }
    \context Voice = "two" {
      \relative c'' {
        g8 g g4 g g
      }
    }
  >>
}
```



```
\score {
  <<
    % score layout
    \new Staff <<
      \context Voice = "one" {
        \voiceOne
      }
      \context Voice = "two" {
        \voiceTwo
      }
    >>

    % musical content
    \context Voice = "one" {
      \relative c'' {
        c4 c c c
      }
    }
```

```
    \context Voice = "two" {
      \relative c'' {
        g8 g g4 g g
      }
    }
  >>
}
```



Alternatively, variables may be employed to similar effect. See Sezione "Organizing pieces with variables" in *Manuale di Apprendimento*.

- `\context` with no name will match the first of any previously created contexts of the same type in the same context heirarchy, even one that has been given a name, and its music expression will be passed to that context for interpretation. This form is rarely useful. However, `\context` with no name and no music expression is used to set the context in which a Scheme procedure specified with `\applyContext` is executed:

```
\new Staff \relative c' {
  c1
  \context Timing
  \applyContext #(lambda (ctx)
                   (newline)
                   (display (ly:context-current-moment ctx)))
  c1
}
```

A context must be named if it is to be referenced later, for example when lyrics are associated with music:

```
\new Voice = "tenor" music
...
\new Lyrics \lyricsto "tenor" lyrics
```

For details of associating lyrics with music see [Automatic syllable durations], pagina 251.

The properties of all contexts of a particular type can be modified in a `\layout` block (with a different syntax), see [Changing all contexts of the same type], pagina 563. This construct also provides a means of keeping layout instructions separate from the musical content. If a single context is to be modified, a `\with` block must be used, see [Changing just one specific context], pagina 566.

Vedi anche

Learning Manual: Sezione "Organizing pieces with variables" in *Manuale di Apprendimento*.

Notation Reference: [Changing just one specific context], pagina 566, [Automatic syllable durations], pagina 251.

5.1.3 Keeping contexts alive

Contexts are usually terminated at the first musical moment in which they have nothing to do. So `Voice` contexts die as soon as they contain no events; `Staff` contexts die as soon as all the `Voice` contexts within them contain no events; etc. This can cause difficulties if earlier contexts which have died have to be referenced, for example, when changing staves with `\change`

commands, associating lyrics with a voice with `\lyricsto` commands, or when adding further musical events to an earlier context.

There is an exception to this general rule: just one of the `Voice` contexts in a `Staff` context or in a `<<...>>` construct will always persist to the end of the enclosing `Staff` context or `<<...>>` construct, even though there may be periods when it has nothing to do. The context to persist in this way will be the first one encountered in the first enclosed `{...}` construct, ignoring any in enclosed `<<...>>` constructs.

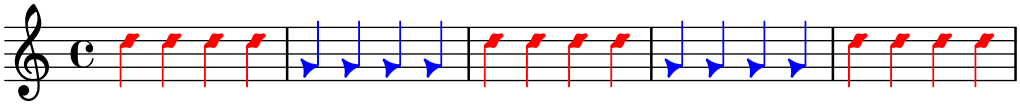
Any context can be kept alive by ensuring it has something to do at every musical moment. `Staff` contexts are kept alive by ensuring one of their voices is kept alive. One way of doing this is to add spacer rests to a voice in parallel with the real music. These need to be added to every `Voice` context which needs to be kept alive. If several voices are to be used sporadically it is safest to keep them all alive rather than attempting to rely on the exceptions mentioned above.

In the following example, both voice A and voice B are kept alive in this way for the duration of the piece:

```
musicA = \relative c'' { d4 d d d }
musicB = \relative c'' { g4 g g g }
keepVoicesAlive = {
  <<
    \new Voice = "A" { s1*5 }  % Keep Voice "A" alive for 5 bars
    \new Voice = "B" { s1*5 }  % Keep Voice "B" alive for 5 bars
  >>
}

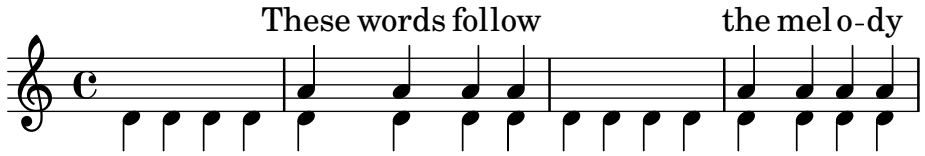
music = {
  \context Voice = "A" {
    \voiceOneStyle
    \musicA
  }
  \context Voice = "B" {
    \voiceTwoStyle
    \musicB
  }
  \context Voice = "A" { \musicA }
  \context Voice = "B" { \musicB }
  \context Voice = "A" { \musicA }
}

\score {
  \new Staff <<
    \keepVoicesAlive
    \music
  >>
}
```



The following example shows how a sporadic melody line with lyrics might be written using this approach. In a real situation the melody and accompaniment would consist of several different sections, of course.

```
melody = \relative c'' { a4 a a a }
accompaniment = \relative c' { d4 d d d }
words = \lyricmode { These words fol -- low the mel -- o -- dy }
\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          s1*4  % Keep Voice "melody" alive for 4 bars
        }
        {
          \new Voice = "accompaniment" {
            \voiceTwo
            \accompaniment
          }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
          \context Voice = "accompaniment" { \accompaniment }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = #"music" }
    \lyricsto "melody" { \words }
  >>
}
```

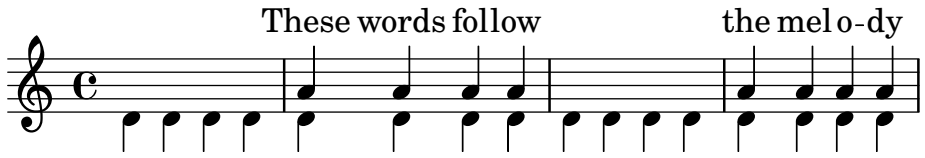


An alternative way, which may be better in many circumstances, is to keep the melody line alive by simply including spacer notes to line it up correctly with the accompaniment:

```
melody = \relative c'' {
  s1  % skip a bar
  a4 a a a
  s1  % skip a bar
  a4 a a a
}
accompaniment = \relative c' {
  d4 d d d
  d4 d d d
  d4 d d d
  d4 d d d
}
```

```
words = \lyricmode { These words fol -- low the mel -- o -- dy }

\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          \melody
        }
        \new Voice = "accompaniment" {
          \voiceTwo
          \accompaniment
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = #"music" }
    \lyricsto "melody" { \words }
  >>
}
```



5.1.4 Modifying context plug-ins

Notation contexts (like `Score` and `Staff`) not only store properties, they also contain plug-ins called 'engravers' that create notation elements. For example, the `Voice` context contains a `Note_heads_engraver` and the `Staff` context contains a `Key_engraver`.

For a full a description of each plug-in, see Internals Reference ↦ Translation ↦ Engravers. Every context described in Internals Reference ↦ Translation ↦ Context. lists the engravers used for that context.
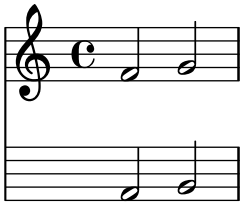
It can be useful to shuffle around these plug-ins. This is done by starting a new context with `\new` or `\context`, and modifying it,

```
\new context \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ...music...
}
```

where the ... should be the name of an engraver. Here is a simple example which removes `Time_signature_engraver` and `Clef_engraver` from a `Staff` context,

```
<<
  \new Staff {
    f2 g
```

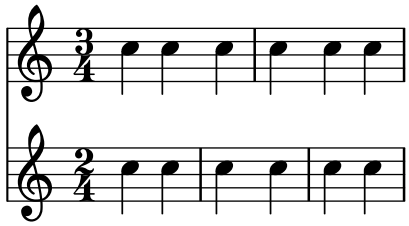
```
  }
  \new Staff \with {
     \remove "Time_signature_engraver"
     \remove "Clef_engraver"
  } {
    f2 g2
  }
>>
```



In the second staff there are no time signature or clef symbols. This is a rather crude method of making objects disappear since it will affect the entire staff. This method also influences the spacing, which may or may not be desirable. More sophisticated methods of blanking objects are shown in Sezione "Visibility and color of objects" in *Manuale di Apprendimento*.

The next example shows a practical application. Bar lines and time signatures are normally synchronized across the score. This is done by the `Timing_translator` and `Default_bar_line_engraver`. This plug-in keeps an administration of time signature, location within the measure, etc. By moving these engraver from `Score` to `Staff` context, we can have a score where each staff has its own time signature.

```
\score {
  <<
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    }
    \relative c'' {
        \time 3/4
        c4 c c c c c
    }
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
  \relative c'' {
      \time 2/4
      c4 c c c c c
  }
>>
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    }
  }
}
```



Problemi noti e avvertimenti

The order in which the engravers are specified is the order in which they are called to carry out their processing. Usually the order in which the engravers are specified does not matter, but in a few special cases the order is important, for example where one engraver writes a property and another reads it, or where one engraver creates a grob and another must process it.

The following orderings are important:

- the `Bar_engraver` must normally be first,
- the `New_fingering_engraver` must come before the `Script_column_engraver`,
- the `Timing_translator` must come before the `Bar_number_engraver`.

Vedi anche

Installed Files: '`ly/engraver-init.ly`'.

5.1.5 Changing context default settings

Context and grob properties can be changed with `\set` and `\override` commands, as described in Sezione 5.3 [Modifying properties], pagina 575. These commands create music events, making the changes take effect at the point in time the music is being processed.

In contrast, this section explains how to change the *default* values of context and grob properties at the time the context is created. There are two ways of doing this. One modifies the default values in all contexts of a particular type, the other modifies the default values in just one particular instance of a context.

Changing all contexts of the same type

The default context settings which are to be used for typesetting in `Score`, `Staff`, `Voice` and other contexts may be specified in a `\context` block within any `\layout` block.

Settings for Midi output as opposed to typesetting will have to be separately specified in `\midi` blocks (see [Output definitions - blueprints for contexts], pagina 553).

The `\layout` block should be placed within the `\score` block to which it is to apply, after the music.

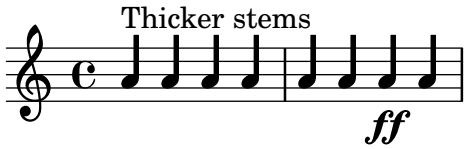
```
\layout {
  \context {
    \Voice
    [context settings for all Voice contexts]
  }
  \context {
    \Staff
    [context settings for all Staff contexts]
  }
}
```

The following types of settings may be specified:

- An `\override` command, but with the context name omitted

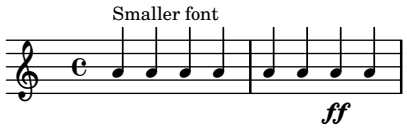
  ```
  \score {
    \relative c'' {
  ```

```
    a4^"Thicker stems" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      \override Stem.thickness = #4.0
    }
  }
}
```



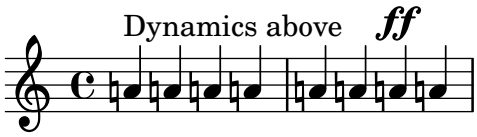
- Directly setting a context property

```
\score {
  \relative c'' {
    a4^"Smaller font" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      fontSize = #-4
    }
  }
}
```



- A predefined command such as `\dynamicUp` or a music expression like `\accidentalStyle dodecaphonic`

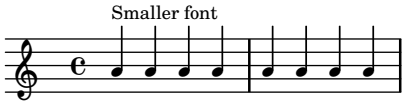
```
\score {
  \relative c'' {
    a4^"Dynamics above" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}
```



- A user-defined variable containing a `\with` block; for details of the `\with` block see [Changing just one specific context], pagina 566.
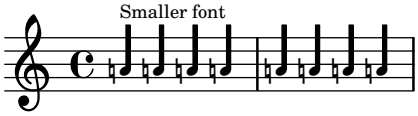
```
StaffDefaults = \with {
  fontSize = #-4
}

\score {
  \new Staff {
    \relative c'' {
      a4^"Smaller font" a a a
      a4 a a a
    }
  }
  \layout {
    \context {
      \Staff
      \StaffDefaults
    }
  }
}
```



Property-setting commands can be placed in a `\layout` block without being enclosed in a `\context` block. Such settings are equivalent to including the same property-setting commands at the start of every context of the type specified. If no context is specified *every* bottom-level context is affected, see [Bottom-level contexts - voices], pagina 555. The syntax of a property-setting command in a `\layout` block is the same as the same command written in the music stream.

```
\score {
  \new Staff {
    \relative c'' {
      a4^"Smaller font" a a a
      a4 a a a
    }
  }
  \layout {
    \accidentalStyle dodecaphonic
    \set fontSize = #-4
    \override Voice.Stem.thickness = #4.0
  }
}
```



Changing just one specific context

The context properties of just one specific context instance can be changed in a `\with` block. All other context instances of the same type retain the default settings built into LilyPond and modified by any `\layout` block within scope. The `\with` block must be placed immediately after the `\new` *context-type* command:

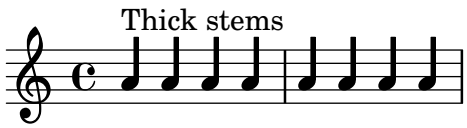
```
\new Staff \with { [context settings for this context instance only] }
{
  ...
}
```

Since such a 'context modification' is specified inside of music, it will affect *all* outputs (typesetting *and* Midi) as opposed to changes within an output definition.

The following types of settings may be specified:

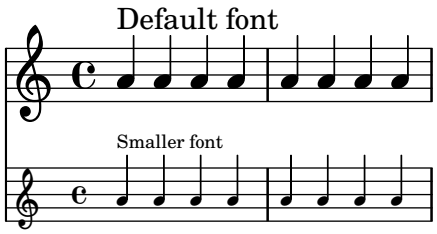
- An `\override` command, but with the context name omitted

```
\score {
  \new Staff {
    \new Voice \with { \override Stem.thickness = #4.0 }
    {
      \relative c'' {
        a4^"Thick stems" a a a
        a4 a a a
      }
    }
  }
}
```



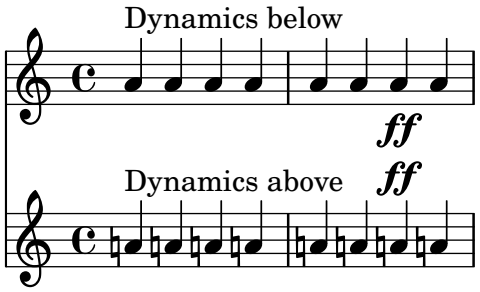
- Directly setting a context property

```
\score {
  <<
    \new Staff {
      \relative c'' {
        a4^"Default font" a a a
        a4 a a a
      }
    }
    \new Staff \with { fontSize = #-4 }
    {
      \relative c'' {
        a4^"Smaller font" a a a
        a4 a a a
      }
    }
  >>
}
```



- A predefined command such as `\dynamicUp`

```
\score {
  <<
    \new Staff {
      \new Voice {
        \relative c'' {
          a4^"Dynamics below" a a a
          a4 a a\ff a
        }
      }
    }
    \new Staff \with { \accidentalStyle dodecaphonic }
    {
      \new Voice \with { \dynamicUp }
      {
        \relative c'' {
          a4^"Dynamics above" a a a
          a4 a a\ff a
        }
      }
    }
  >>
}
```



Order of precedence

The value of a property which applies at a particular time is determined as follows:

- if an `\override` or `\set` command in the input stream is in effect that value is used,
- otherwise the default value taken from a `\with` statement on the context initiation statement is used,
- otherwise the default value taken from the most recent appropriate `\context` block in the `\layout` or `\midi` blocks is used,
- otherwise the LilyPond built-in default is used.

Vedi anche

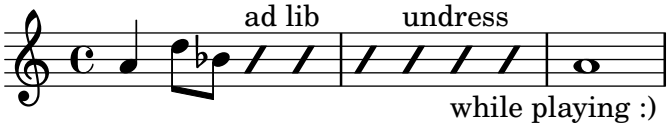
Learning Manual: Sezione “Modifying context properties” in *Manuale di Apprendimento*.

Notation Reference: Sezione 5.1.1 [Contexts explained], pagina 553, [Bottom-level contexts - voices], pagina 555, Sezione 5.3.2 [The set command], pagina 575, Sezione 5.3.3 [The override command], pagina 577, Sezione 4.2.1 [The `\layout` block], pagina 512.

5.1.6 Defining new contexts

Specific contexts, like `Staff` and `Voice`, are made from simple building blocks. It is possible to create new types of contexts with different combinations of engraver plug-ins.

The next example shows how to build a different type of `Voice` context from scratch. It will be similar to `Voice`, but only prints centered slash note heads. It can be used to indicate improvisation in jazz pieces,



These settings are defined within a `\context` block inside a `\layout` block,

```
\layout {
  \context {
    ...
  }
}
```

In the following discussion, the example input shown should go in place of the ... in the previous fragment.

First it is necessary to define a name for the new context:

```
\name ImproVoice
```

Since it is similar to the `Voice` context, we want commands that work in (existing) `Voice` contexts to continue working. This is achieved by giving the new context an alias of `Voice`,

```
\alias Voice
```

The context will print notes and instructive texts, so we need to add the engravers which provide this functionality, plus the engraver which groups notes, stems and rests which occur at the same musical moment into columns,

```
\consists "Note_heads_engraver"
\consists "Text_engraver"
\consists "Rhythmic_column_engraver"
```

The note heads should all be placed on the center line,

```
\consists "Pitch_squash_engraver"
squashedPosition = #0
```

The `Pitch_squash_engraver` modifies note heads (created by the `Note_heads_engraver`) and sets their vertical position to the value of `squashedPosition`, in this case `0`, the center line.

The notes look like a slash, and have no stem,

```
\override NoteHead.style = #'slash
\hide Stem
```

All these plug-ins have to communicate under the control of the context. The mechanisms with which contexts communicate are established by declaring the context `\type`. Within a `\layout` block, most contexts will be of type `Engraver_group`. Some special contexts and contexts in `\midi` blocks use other context types. Copying and modifying an existing context definition will also fill in the type. Since this example creates a definition from scratch, it needs to be specified explicitly.

```
\type "Engraver_group"
```

Put together, we get

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists "Rhythmic_column_engraver"
  \consists "Pitch_squash_engraver"
  squashedPosition = #0
  \override NoteHead.style = #'slash
  \hide Stem
  \alias Voice
}
```

Contexts form hierarchies. We want to place the `ImproVoice` context within the `Staff` context, just like normal `Voice` contexts. Therefore, we modify the `Staff` definition with the `\accepts` command,

```
\context {
  \Staff
  \accepts ImproVoice
}
```

The opposite of `\accepts` is `\denies`, which is sometimes needed when reusing existing context definitions.

Putting both into a `\layout` block, like

```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \context {
    \Staff
    \accepts "ImproVoice"
  }
}
```

Then the output at the start of this subsection can be entered as

```
\relative c'' {
  a4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"undress"
    c c_"while playing :)"
  }
  a1
}
```

To complete this example, changes affecting the context hierarchy should be repeated in a `\midi` block so that Midi output depends on the same context relations.

Vedi anche

Internals Reference: Sezione “Engraver_group” in *Guida al Funzionamento Interno*, Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Text_engraver” in *Guida al Funzionamento Interno*, Sezione “Rhythmic_column_engraver” in *Guida al Funzionamento Interno*, Sezione “Pitch_squash_engraver” in *Guida al Funzionamento Interno*.

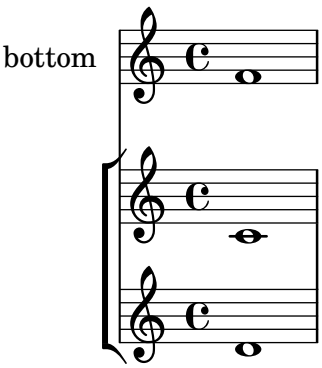
5.1.7 Context layout order

Contexts are normally positioned in a system from top to bottom in the order in which they are encountered in the input file. When contexts are nested, the outer context will include inner nested contexts as specified in the input file, provided the inner contexts are included in the outer context’s “accepts” list. Nested contexts which are not included in the outer context’s “accepts” list will be repositioned below the outer context rather than nested within it.

The “accepts” list of a context can be changed with the `\accepts` or `\denies` commands. `\accepts` adds a context to the “accepts” list and `\denies` removes a context from the list.

For example, a square-braced staff group is not usually found within a curved-braced staff with connecting staff bars, and a `GrandStaff` does not accept a `StaffGroup` inside it by default.

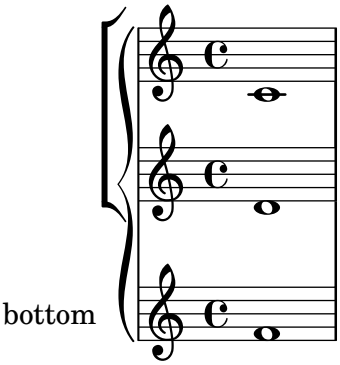
```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = bottom f'1 }
  >>
}
```



However, by using the `\accepts` command, `StaffGroup` can be added to the `GrandStaff` context:

```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = bottom f'1 }
  >>
  \layout {
    \context {
      \GrandStaff
      \accepts "StaffGroup"
    }
```

```
  }
}
```



`\denies` is mainly used when a new context is being based on another, but the required nesting differs. For example, the `VaticanaStaff` context is based on the `Staff` context, but with the `VaticanaVoice` context substituted for the `Voice` context in the "accepts" list.

Note that a context will be silently created implicitly if a command is encountered when there is no suitable context available to contain it.

Within a context definition, the type of subcontext to be implicitly created is specified using `\defaultchild`. A number of music events require a '`Bottom`' context: when such an event is encountered, subcontexts are created recursively until reaching a context with no '`defaultchild`' setting.

Implicit context creation can at times give rise to unexpected new staves or scores. Using `\new` to create contexts explicitly avoids those problems.

Sometimes a context is required to exist for just a brief period, a good example being the staff context for an ossia. This is usually achieved by introducing the context definition at the appropriate place in parallel with corresponding section of the main music. By default, the temporary context will be placed below all the existing contexts. To reposition it above the context called "main", it should be defined like this:

```
\new Staff \with { alignAboveContext = #"main" }
```

A similar situation arises when positioning a temporary lyrics context within a multi-staved layout such as a `ChoirStaff`, for example, when adding a second verse to a repeated section. By default the temporary lyrics context will be placed beneath the lower staves. By defining the temporary lyrics context with `alignBelowContext` it can be positioned correctly beneath the (named) lyrics context containing the first verse.

Examples showing this repositioning of temporary contexts can be found elsewhere — see Sezione "Nesting music expressions" in *Manuale di Apprendimento*, ⟨undefined⟩ [Modifying single staves], pagina ⟨undefined⟩ and Sezione 2.1.2 [Techniques specific to lyrics], pagina 259.

Vedi anche

Learning Manual: Sezione "Nesting music expressions" in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Modifying single staves], pagina ⟨undefined⟩, Sezione 2.1.2 [Techniques specific to lyrics], pagina 259.

Application Usage: Sezione "An extra staff appears" in *Uso del Programma*.
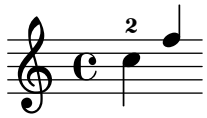
Installed Files: '`ly/engraver-init.ly`'.

5.2 Explaining the Internals Reference

5.2.1 Navigating the program reference

Suppose we want to move the fingering indication in the fragment below:

```
c-2
\stemUp
f
```



If you visit the documentation on fingering instructions (in ⟨undefined⟩ [Fingering instructions], pagina ⟨undefined⟩), you will notice:

See also

Internals Reference: Sezione “Fingering” in *Guida al Funzionamento Interno*.

The programmer’s reference is available as an HTML document. It is highly recommended that you read it in HTML form, either online or by downloading the HTML documentation. This section will be much more difficult to understand if you are using the PDF manual.

Follow the link to Sezione “Fingering” in *Guida al Funzionamento Interno*. At the top of the page, you will see

Fingering objects are created by: Sezione “Fingering_engraver” in *Guida al Funzionamento Interno* and Sezione “New_fingering_engraver” in *Guida al Funzionamento Interno*.

By following related links inside the program reference, we can follow the flow of information within the program:

- Sezione “Fingering” in *Guida al Funzionamento Interno*: Sezione “Fingering” in *Guida al Funzionamento Interno* objects are created by: Sezione “Fingering_engraver” in *Guida al Funzionamento Interno*
- Sezione “Fingering_engraver” in *Guida al Funzionamento Interno*: Music types accepted: Sezione “fingering-event” in *Guida al Funzionamento Interno*
- Sezione “fingering-event” in *Guida al Funzionamento Interno*: Music event type `fingering-event` is in Music expressions named Sezione “FingeringEvent” in *Guida al Funzionamento Interno*

This path goes against the flow of information in the program: it starts from the output, and ends at the input event. You could also start at an input event, and read with the flow of information, eventually ending up at the output object(s).

The program reference can also be browsed like a normal document. It contains chapters on `Music definitions` on Sezione “Translation” in *Guida al Funzionamento Interno*, and the Sezione “Backend” in *Guida al Funzionamento Interno*. Every chapter lists all the definitions used and all properties that may be tuned.

5.2.2 Layout interfaces

The HTML page that we found in the previous section describes the layout object called Sezione “Fingering” in *Guida al Funzionamento Interno*. Such an object is a symbol within the score. It has properties that store numbers (like thicknesses and directions), but also pointers to related objects. A layout object is also called a *Grob*, which is short for Graphical Object. For more details about Grobs, see Sezione “grob-interface” in *Guida al Funzionamento Interno*.

The page for `Fingering` lists the definitions for the `Fingering` object. For example, the page says

`padding` (dimension, in staff space):

`0.5`

which means that the number will be kept at a distance of at least 0.5 of the note head.

Each layout object may have several functions as a notational or typographical element. For example, the Fingering object has the following aspects

- Its size is independent of the horizontal spacing, unlike slurs or beams.
- It is a piece of text. Granted, it is usually a very short text.
- That piece of text is typeset with a font, unlike slurs or beams.
- Horizontally, the center of the symbol should be aligned to the center of the note head.
- Vertically, the symbol is placed next to the note and the staff.
- The vertical position is also coordinated with other superscript and subscript symbols.

Each of these aspects is captured in so-called *interface*s, which are listed on the Sezione "Fingering" in *Guida al Funzionamento Interno* page at the bottom

This object supports the following interfaces: Sezione "item-interface" in *Guida al Funzionamento Interno*, Sezione "self-alignment-interface" in *Guida al Funzionamento Interno*, Sezione "side-position-interface" in *Guida al Funzionamento Interno*, Sezione "text-interface" in *Guida al Funzionamento Interno*, Sezione "text-script-interface" in *Guida al Funzionamento Interno*, Sezione "font-interface" in *Guida al Funzionamento Interno*, Sezione "finger-interface" in *Guida al Funzionamento Interno*, and Sezione "grob-interface" in *Guida al Funzionamento Interno*.

Clicking any of the links will take you to the page of the respective object interface. Each interface has a number of properties. Some of them are not user-serviceable ('Internal properties'), but others can be modified.

We have been talking of *the* `Fingering` object, but actually it does not amount to much. The initialization file (see Sezione "Other sources of information" in *Manuale di Apprendimento*) '`scm/define-grobs.scm`' shows the soul of the 'object',
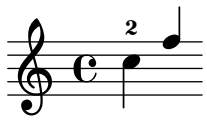
```
(Fingering
  . ((padding . 0.5)
     (avoid-slur . around)
     (slur-padding . 0.2)
     (staff-padding . 0.5)
     (self-alignment-X . 0)
     (self-alignment-Y . 0)
     (script-priority . 100)
     (stencil . ,ly:text-interface::print)
     (direction . ,ly:script-interface::calc-direction)
     (font-encoding . fetaText)
     (font-size . -5)  ; don't overlap when next to heads.
     (meta . ((class . Item)
     (interfaces . (finger-interface
                    font-interface
                    text-script-interface
                    text-interface
                    side-position-interface
                    self-alignment-interface
                    item-interface))))))
```

As you can see, the `Fingering` object is nothing more than a bunch of variable settings, and the webpage in the Internals Reference is directly generated from this definition.

5.2.3 Determining the grob property

Recall that we wanted to change the position of the **2** in

```
c-2
\stemUp
f
```



Since the **2** is vertically positioned next to its note, we have to meddle with the interface associated with this positioning. This is done using `side-position-interface`. The page for this interface says

`side-position-interface`

Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

Below this description, the variable `padding` is described as

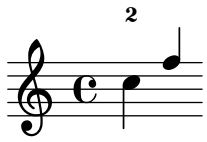
`padding` (dimension, in staff space)
Add this much extra space between objects that are next to each other.

By increasing the value of `padding`, we can move the fingering away from the note head. The following command inserts 3 staff spaces of white between the note and the fingering:

```
\once \override Voice.Fingering.padding = #3
```

Inserting this command before the Fingering object is created, i.e., before `c2`, yields the following result:

```
\once \override Voice.Fingering.padding = #3
c-2
\stemUp
f
```



In this case, the context for this tweak is `Voice`. This fact can also be deduced from the program reference, for the page for the Sezione “Fingering_engraver” in *Guida al Funzionamento Interno* plug-in says

Fingering_engraver is part of contexts: . . . Sezione “Voice” in *Guida al Funzionamento Interno*

5.2.4 Naming conventions

Another thing that is needed, is an overview of the various naming conventions:

- scheme functions: lowercase-with-hyphens (incl. one-word names)
- scheme functions: ly:plus-scheme-style
- music events, music classes and music properties: as-scheme-functions
- Grob interfaces: scheme-style
- backend properties: scheme-style (but X and Y!)
- contexts (and MusicExpressions and grobs): Capitalized or CamelCase

- context properties: lowercaseFollowedByCamelCase
- engravers: Capitalized_followed_by_lowercase_and_with_underscores

Questions to be answered:

- Which of these are conventions and which are rules?
- Which are rules of the underlying language, and which are LP-specific?

5.3 Modifying properties

5.3.1 Overview of modifying properties

Each context is responsible for creating certain types of graphical objects. The settings used for printing these objects are also stored by context. By changing these settings, the appearance of objects can be altered.

There are two different kinds of properties stored in contexts: context properties and grob properties. Context properties are properties that apply to the context as a whole and control how the context itself is displayed. In contrast, grob properties apply to specific grob types that will be displayed in the context.

The `\set` and `\unset` commands are used to change values for context properties. The `\override` and `\revert` commands are used to change values for grob properties.

Vedi anche

Internals Reference: Sezione "Backend" in *Guida al Funzionamento Interno*, Sezione "All layout objects" in *Guida al Funzionamento Interno*, Sezione "OverrideProperty" in *Guida al Funzionamento Interno*, Sezione "RevertProperty" in *Guida al Funzionamento Interno*, Sezione "PropertySet" in *Guida al Funzionamento Interno*.

Problemi noti e avvertimenti

The back-end is not very strict in type-checking object properties. Cyclic references in Scheme values for properties can cause hangs or crashes, or both.

5.3.2 The `\set` command

Each context has a set of *properties*, variables contained in that context. Context properties are changed with the `\set` command, which has the following syntax:

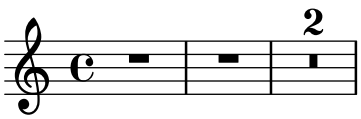
```
\set context.property = #value
```

value is a Scheme object, which is why it must be preceded by the `#` character.

Contexts properties are usually named in `studlyCaps`. They mostly control the translation from music to notation, e.g. `localKeySignature` (for determining whether to print accidentals), or `measurePosition` (for determining when to print a bar line). Context properties can change value over time while interpreting a piece of music; `measurePosition` is an obvious example of this. Context properties are modified with `\set`.
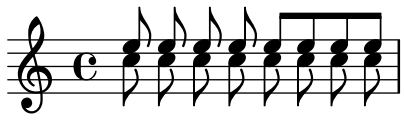
For example, multimeasure rests will be combined into a single bar if the context property `skipBars` is set to `#t`:

```
R1*2
\set Score.skipBars = ##t
R1*2
```



If the *context* argument is left out, then the property will be set in the current bottom context (typically `ChordNames`, `Voice`, `TabVoice`, or `Lyrics`).
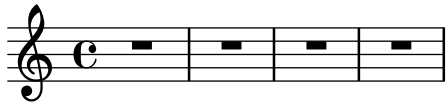
```
\set Score.autoBeaming = ##f
<<
  {
    e8 e e e
    \set autoBeaming = ##t
    e8 e e e
  } \\ {
    c8 c c c c8 c c c
  }
>>
```



The change is applied 'on-the-fly', during the music, so that the setting only affects the second group of eighth notes.

Note that the bottom-most context does not always contain the property that you wish to change – for example, attempting to set the `skipBars` property of the default bottom context, in this case `Voice`, will have no effect, because skipBars is a property of the `Score` context.

```
R1*2
\set skipBars = ##t
R1*2
```



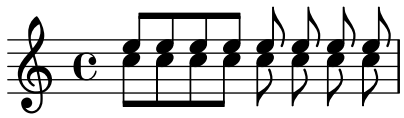
Contexts are hierarchical, so if an enclosing context was specified, for example `Staff`, then the change would also apply to all `Voice`s in the current staff.

The `\unset` command:

`\unset` *`context.property`*

is used to remove the definition of *property* from *context*. This command removes the definition only if it is set in *context*. Properties that have been set in enclosing contexts will not be altered by an unset in an enclosed context:

```
\set Score.autoBeaming = ##t
<<
  {
    \unset autoBeaming
    e8 e e e
    \unset Score.autoBeaming
    e8 e e e
  } \\ {
    c8 c c c c8 c c c
  }
>>
```
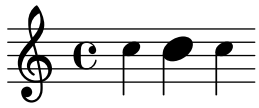


Like `\set`, the *context* argument does not have to be specified for a bottom context, so the two statements

```
\set Voice.autoBeaming = ##t
\set autoBeaming = ##t
```

are equivalent if the current bottom context is `Voice`.

Preceding a `\set` command by `\once` makes the setting apply to only a single time-step:

```
c4
\once \set fontSize = #4.7
c4
c4
```



A full description of all available context properties is in the internals reference, see Translation ↦ Tunable context properties.

Vedi anche

Internals Reference: Sezione "Tunable context properties" in *Guida al Funzionamento Interno*.

5.3.3 The `\override` command
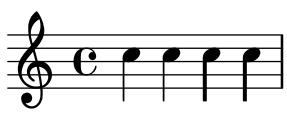
There is a special type of context property: the grob description. Grob descriptions are named in `StudlyCaps` (starting with capital letters). They contain the 'default settings' for a particular kind of grob as an association list. See '`scm/define-grobs.scm`' to see the settings for each grob description. Grob descriptions are modified with `\override`.

The syntax for the `\override` command is

```
\override [context.]GrobName.property = #value
```

For example, we can increase the thickness of a note stem by overriding the `thickness` property of the `Stem` object:
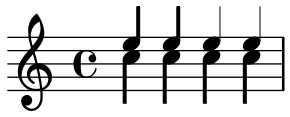
```
c4 c
\override Voice.Stem.thickness = #3.0
c4 c
```



If no context is specified in an `\override`, the bottom context is used:

```
{ \override Staff.Stem.thickness = #3.0
  <<
    {
      e4 e
      \override Stem.thickness = #0.5
      e4 e
    } \\ {
      c4 c c c
```

```
    }
  >>
}
```



Some tweakable options are called 'subproperties' and reside inside properties. To tweak those, use commands in the form

```
\override Stem.details.beamed-lengths = #'(4 4 3)
```

or to modify the ends of spanners, use a form like these

```
\override TextSpanner.bound-details.left.text = #"left text"
\override TextSpanner.bound-details.right.text = #"right text"
```
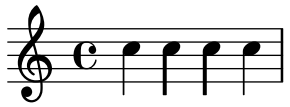
The effects of `\override` can be undone by `\revert`.

The syntax for the `\revert` command is

```
\revert [context.]GrobName.property
```
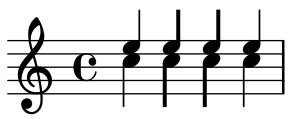
For example,

```
c4
\override Voice.Stem.thickness = #3.0
c4 c
\revert Voice.Stem.thickness
c4
```
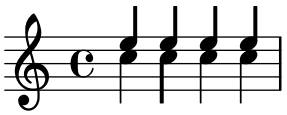


The effects of `\override` and `\revert` apply to all grobs in the affected context from the current time forward:

```
{
  <<
    {
      e4
      \override Staff.Stem.thickness = #3.0
      e4 e e
    } \\ {
      c4 c c
      \revert Staff.Stem.thickness
      c4
    }
  >>
}
```



`\once` can be used with `\override` to affect only the current time step:

```
{
  <<
    {
      \override Stem.thickness = #3.0
      e4 e e e
    } \\ {
      c4
      \once \override Stem.thickness = #3.0
      c4 c c
    }
  >>
}
```



Vedi anche

Internals Reference: Sezione “Backend” in *Guida al Funzionamento Interno*

5.3.4 The `\tweak` command

Changing grob properties with `\override` causes the changes to apply to all of the given grobs in the context at the moment the change applies. Sometimes, however, it is desirable to have changes apply to just one grob, rather than to all grobs in the affected context. This is accomplished with the `\tweak` command, which has the following syntax:

`\tweak [`*layout-object*`.]`*grob-property value*

Specifying *layout-object* is optional. The `\tweak` command applies to the music object that immediately follows *value* in the music stream.
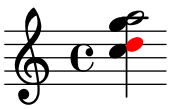
For an introduction to the syntax and uses of the tweak command see Sezione “Tweaking methods” in *Manuale di Apprendimento*.

When several similar items are placed at the same musical moment, the `\override` command cannot be used to modify just one of them – this is where the `\tweak` command must be used. Items which may appear more than once at the same musical moment include the following:

- note heads of notes inside a chord
- articulation signs on a single note
- ties between notes in a chord
- tuplet brackets starting at the same time
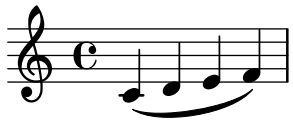
In this example, the color of one note head and the type of another note head are modified within a single chord:

```
< c
  \tweak color #red
  d
  g
  \tweak duration-log #1
  a
> 4
```
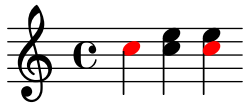


`\tweak` can be used to modify slurs:

```
c-\tweak thickness #5 ( d e f)
```



For the `\tweak` command to work, it must remain immediately adjacent to the object to which it is to apply after the input file has been converted to a music stream. Tweaking a whole chord does not do anything since its music event only acts as a container, and all layout objects are created from events inside of the `EventChord`:
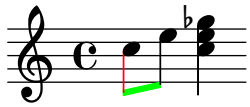
```
\tweak color #red c4
\tweak color #red <c e>4
<\tweak color #red c e>4
```



The simple `\tweak` command cannot be used to modify any object that is not directly created from the input. In particular it will not affect stems, automatic beams or accidentals, since these are generated later by `NoteHead` layout objects rather than by music elements in the input stream.

Such indirectly created layout objects can be tweaked using the form of the `\tweak` command in which the grob name is specified explicitly:
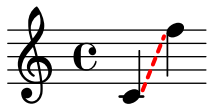
```
\tweak Stem.color #red
\tweak Beam.color #green c8 e
<c e \tweak Accidental.font-size #-3 ges>4
```



`\tweak` cannot be used to modify clefs or time signatures, since these become separated from any preceding `\tweak` command in the input stream by the automatic insertion of extra elements required to specify the context.

Several `\tweak` commands may be placed before a notational element – all affect it:

```
c
-\tweak style #'dashed-line
-\tweak dash-fraction #0.2
-\tweak thickness #3
-\tweak color #red
 \glissando
f'
```



The music stream which is generated from a section of an input file, including any automatically inserted elements, may be examined, see Sezione “Displaying music expressions” in *Estendere*. This may be helpful in determining what may be modified by a `\tweak` command, or in determining how to adjust the input to make a `\tweak` apply.

Vedi anche

Learning Manual: Sezione “Tweaking methods” in *Manuale di Apprendimento*.

Extending LilyPond: Sezione “Displaying music expressions” in *Estendere*.

Problemi noti e avvertimenti

The `\tweak` command cannot be used to modify the control points of just one of several ties in a chord, other than the first one encountered in the input file.

5.3.5 `\set` vs. `\override`

Both `\set` and `\override` manipulate properties associated with contexts. In either case, properties heed the hierarchy of contexts: properties not set in a context itself show the values of the respective parent context.

Values and lifetime of context properties are dynamic and only available when music is being interpreted, ‘iterated’. At the time of context creation, properties are initialized from the corresponding context definition and possible context modifications. Afterwards, changes are achieved with property-setting commands in the music itself.

Now grob definitions are a special category of context properties. Since their structure, bookkeeping and use is different from ordinary context properties, they are accessed with a different set of commands, and treated separately in the documentation.

As opposed to plain context properties, grob definitions are subdivided into grob properties. A “grob” (graphical object) is usually created by an engraver at the time of interpreting a music expression and receives its initial properties from the current grob definition of the engraver’s context. The engraver (or other ‘backend’ parts of LilyPond) may subsequently add or change properties to the grob, but that does not affect the context’s grob definition.

What we call ‘grob properties’ in the context of user-level tweaking are actually the properties of a context’s grob definition. In contrast to ordinary context properties, grob definitions have the bookkeeping required to keep track of its parts, the individual grob properties (and even subproperties of them) separately so that it is possible to define those parts in different contexts and have the overall grob definition at the time of grob creation be assembled from pieces provided in different contexts among the current context and its parents.

Grob definitions are manipulated using `\override` and `\revert` and have a name starting with a capital letter (like ‘`NoteHead`’) whereas ordinary context properties are manipulated using `\set` and `\unset` and are named starting with a lowercase letter.

The special commands `\tweak` and `\overrideProperty` change grob properties bypassing context properties completely. Instead they catch grobs as they are being created and then directly set properties on them when they originate from a tweaked music event or are of a particular kind, respectively.

5.3.6 Modifying alists

Some user-configurable properties are internally represented as *alists* (association lists), which store pairs of *keys* and *values*. The structure of an alist is:

```
'((key1 . value1)
  (key2 . value2)
  (key3 . value3)
  ...)
```

If an alist is a grob property or `\paper` variable, its keys can be modified individually without affecting other keys.

For example, to reduce the space between adjacent staves in a staff-group, use the `staff-staff-spacing` property of the `StaffGrouper` grob. The property is an alist with four

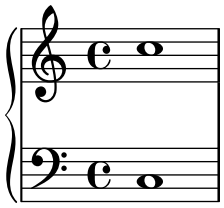
keys: `basic-distance`, `minimum-distance`, `padding`, and `stretchability`. The standard settings for this property are listed in the "Backend" section of the Internals Reference (see Sezione "StaffGrouper" in *Guida al Funzionamento Interno*):

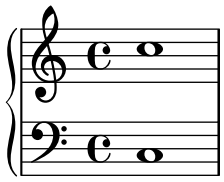
```
'((basic-distance . 9)
  (minimum-distance . 7)
  (padding . 1)
  (stretchability . 5))
```

One way to bring the staves closer together is by reducing the value of the `basic-distance` key (`9`) to match the value of `minimum-distance` (`7`). To modify a single key individually, use a *nested declaration*:

```
% default space between staves
\new PianoStaff <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>

% reduced space between staves
\new PianoStaff \with {
  % this is the nested declaration
  \override StaffGrouper.staff-staff-spacing.basic-distance = #7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```
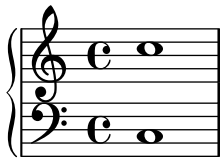




Using a nested declaration will update the specified key (such as `basic-distance` in the above example) without altering any other keys already set for the same property.

Now suppose we want the staves to be as close as possible without overlapping. The simplest way to do this is to set all four alist keys to zero. However, it is not necessary to enter four nested declarations, one for each key. Instead, the property can be completely re-defined with one declaration, as an alist:

```
\new PianoStaff \with {
  \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 0)
       (minimum-distance . 0)
       (padding . 0)
       (stretchability . 0))
} <<
```

```
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



Note that any keys not explicitly listed in the alist definition will be reset to their *default-when-unset* values. In the case of `staff-staff-spacing`, any unset key-values would be reset to zero (except `stretchability`, which takes the value of `basic-distance` when unset). Thus the following two declarations are equivalent:

```
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7))

\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7)
     (minimum-distance . 0)
     (padding . 0)
     (stretchability . 7))
```

One (possibly unintended) consequence of this is the removal of any standard settings that are set in an initialization file and loaded each time an input file is compiled. In the above example, the standard settings for `padding` and `minimum-distance` (defined in '`scm/define-grobs.scm`') are reset to their default-when-unset values (zero for both keys). Defining a property or variable as an alist (of any size) will always reset all unset key-values to their default-when-unset values. Unless this is the intended result, it is safer to update key-values individually with a nested declaration.

Nota: Nested declarations will not work for context property alists (such as `beamExceptions`, `keySignature`, `timeSignatureSettings`, etc.). These properties can only be modified by completely re-defining them as alists.

5.4 Useful concepts and properties

5.4.1 Input modes

The way in which the notation contained within an input file is interpreted is determined by the current input mode.

Chord mode

This is activated with the `\chordmode` command, and causes input to be interpreted with the syntax of chord notation, see Sezione 2.7 [Chord notation], pagina 392. Chords are rendered as notes on a staff.

Chord mode is also activated with the `\chords` command. This also creates a new `ChordNames` context and causes the following input to be interpreted with the syntax of chord notation and rendered as chord names in the `ChordNames` context, see [Printing chord names], pagina 398.

Drum mode

This is activated with the `\drummode` command, and causes input to be interpreted with the syntax of drum notation, see [Basic percussion notation], pagina 370.

Drum mode is also activated with the `\drums` command. This also creates a new `DrumStaff` context and causes the following input to be interpreted with the syntax of drum notation and rendered as drum symbols on a drum staff, see [Basic percussion notation], pagina 370.

Figure mode

This is activated with the `\figuremode` command, and causes input to be interpreted with the syntax of figured bass, see [Entering figured bass], pagina 407.

Figure mode is also activated with the `\figures` command. This also creates a new `FiguredBass` context and causes the following input to be interpreted with the figured bass syntax and rendered as figured bass symbols in the `FiguredBass` context, see [Introduction to figured bass], pagina 406.

Fret and tab modes

There are no special input modes for entering fret and tab symbols.

To create tab diagrams, enter notes or chords in note mode and render them in a `TabStaff` context, see [Default tablatures], pagina 326.

To create fret diagrams above a staff, you have two choices. You can either use the `FretBoards` context (see [Automatic fret diagrams], pagina 361 or you can enter them as a markup above the notes using the `\fret-diagram` command (see [Fret diagram markups], pagina 342).

Lyrics mode

This is activated with the `\lyricmode` command, and causes input to be interpreted as lyric syllables with optional durations and associated lyric modifiers, see Sezione 2.1 [Vocal music], pagina 247.

Lyric mode is also activated with the `\addlyrics` command. This also creates a new `Lyrics` context and an implicit `\lyricsto` command which associates the following lyrics with the preceding music.

Markup mode

This is activated with the `\markup` command, and causes input to be interpreted with the syntax of markup, see Sezione A.10 [Text markup commands], pagina 653.
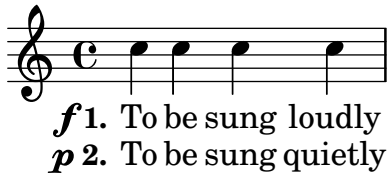
Note mode

This is the default mode or it may be activated with the `\notemode` command. Input is interpreted as pitches, durations, markup, etc and typeset as musical notation on a staff.

It is not normally necessary to specify note mode explicitly, but it may be useful to do so in certain situations, for example if you are in lyric mode, chord mode or any other mode and want to insert something that only can be done with note mode syntax.

For example, to indicate dynamic markings for the verses of a choral pieces it is necessary to enter note mode to interpret the markings:

```
{ c4 c4 c4 c4 }
\addlyrics {
  \notemode{\set stanza = \markup{ \dynamic f 1. } }
  To be sung loudly
}
\addlyrics {
  \notemode{\set stanza = \markup{ \dynamic p 2. } }
  To be sung quietly
}
```



5.4.2 Direction and placement

In typesetting music the direction and placement of many items is a matter of choice. For example, the stems of notes can be directed up or down; lyrics, dynamics, and other expressive marks may be placed above or below the staff; text may be aligned left, right or center; etc. Most of these choices may be left to be determined automatically by LilyPond, but in some cases it may be desirable to force a particular direction or placement.

Articulation direction indicators

By default some directions are always up or always down (e.g. dynamics or fermata), while other things can alternate between up or down based on the stem direction (like slurs or accents).

The default action may be overridden by prefixing the articulation by a *direction indicator*. Three direction indicators are available: `^` (meaning "up"), `_` (meaning "down") and `-` (meaning "use default direction"). The direction indicator can usually be omitted, in which case `-` is assumed, but a direction indicator is **always** required before

- `\tweak` commands
- `\markup` commands
- `\tag` commands
- string markups, e.g. `-"string"`
- fingering instructions, e.g. `-1`
- articulation shortcuts, e.g. `-.`, `->`, `--`

Direction indicators affect only the next note:

```
c2( c)
c2_( c)
c2( c)
c2^( c)
```



The direction property

The position or direction of many layout objects is controlled by the `direction` property.

The value of the `direction` property may be set to `1`, meaning "up" or "above", or to `-1`, meaning "down" or "below". The symbols `UP` and `DOWN` may be used instead of `1` and `-1` respectively. The default direction may be specified by setting `direction` to `0` or `CENTER`. Alternatively, in many cases predefined commands exist to specify the direction. These are of the form
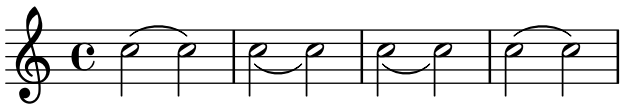
```
\xxxUp, \xxxDown or \xxxNeutral
```

where `\xxxNeutral` means "use the default" direction. See Sezione "Within-staff objects" in *Manuale di Apprendimento*.

In a few cases, arpeggio for example, the value of the `direction` property can specify whether the object is to be placed to the right or left of the parent. In this case `-1` or `LEFT` means "to the left" and `1` or `RIGHT` means "to the right". `0` or `CENTER` means "use the default" direction.

These indications affect all notes until they are canceled.

```
c2( c)
\slurDown
c2( c)
c2( c)
\slurNeutral
c2( c)
```



In polyphonic music, it is generally better to specify an explicit `voice` than change an object's direction. For more information. See ⟨undefined⟩ [Multiple voices], pagina ⟨undefined⟩.

Vedi anche

Learning Manual: Sezione "Within-staff objects" in *Manuale di Apprendimento*.

Notation Reference: ⟨undefined⟩ [Multiple voices], pagina ⟨undefined⟩.

5.4.3 Distances and measurements

Distances in LilyPond are of two types: absolute and scaled.

Absolute distances are used for specifying margins, indents, and other page layout details, and are by default specified in millimeters. Distances may be specified in other units by following the quantity by `\mm`, `\cm`, `\in` (inches), or `\pt` (points, 1/72.27 of an inch). Page layout distances can also be specified in scalable units (see the following paragraph) by appending `\staff-space` to the quantity. Page layout is described in detail in Sezione 4.1 [Page layout], pagina 502.

Scaled distances are always specified in units of the staff-space or, rarely, the half staff-space. The staff-space is the distance between two adjacent staff lines. The default value can be changed globally by setting the global staff size, or it can be overridden locally by changing the `staff-space` property of `StaffSymbol`. Scaled distances automatically scale with any change to the either the global staff size or the `staff-space` property of `StaffSymbol`, but fonts scale automatically only with changes to the global staff size. The global staff size thus enables the overall size of a rendered score to be easily varied. For the methods of setting the global staff size see Sezione 4.2.2 [Setting the staff size], pagina 514.

If just a section of a score needs to be rendered to a different scale, for example an ossia section or a footnote, the global staff size cannot simply be changed as this would affect the entire score. In such cases the change in size is made by overriding both the `staff-space` property of `StaffSymbol` and the size of the fonts. A Scheme function, `magstep`, is available to convert from a font size change to the equivalent change in `staff-space`. For an explanation and an example of its use, see Sezione "Length and thickness of objects" in *Manuale di Apprendimento*.

Vedi anche

Learning Manual: Sezione "Length and thickness of objects" in *Manuale di Apprendimento*.
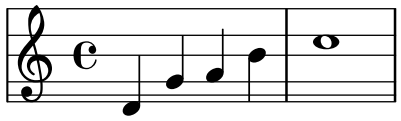
Notation Reference: Sezione 4.1 [Page layout], pagina 502, Sezione 4.2.2 [Setting the staff size], pagina 514.

5.4.4 Staff symbol properties

The vertical position of staff lines and the number of staff lines can be defined at the same time. As the following example shows, note positions are not influenced by the staff line positions.
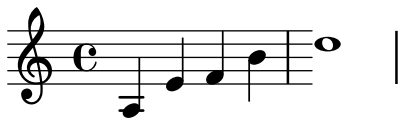
Nota: The `'line-positions` property overrides the `'line-count` property. The number of staff lines is implicitly defined by the number of elements in the list of values for `'line-positions`.

```
\new Staff \with {
  \override StaffSymbol.line-positions = #'(7 3 0 -4 -6 -7)
}
{ a4 e' f b | d1 }
```



The width of a staff can be modified. The units are staff spaces. The spacing of objects inside the staff is not affected by this setting.

```
\new Staff \with {
  \override StaffSymbol.width = #23
}
{ a4 e' f b | d1 }
```



5.4.5 Spanners

Many objects of musical notation extend over several notes or even several bars. Examples are slurs, beams, tuplet brackets, volta repeat brackets, crescendi, trills, and glissandi. Such objects are collectively called "spanners", and have special properties to control their appearance and behaviour. Some of these properties are common to all spanners; others are restricted to a sub-set of the spanners.

All spanners support the `spanner-interface`. A few, essentially those that draw a straight line between the two objects, support in addition the `line-spanner-interface`.

Using the `spanner-interface`

This interface provides two properties that apply to several spanners.

The `minimum-length` *property*

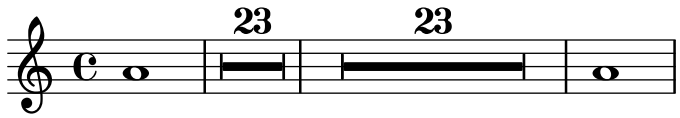
The minimum length of the spanner is specified by the `minimum-length` property. Increasing this usually has the necessary effect of increasing the spacing of the notes between the two end points. However, this override has no effect on many spanners, as their length is determined by other considerations. A few examples where it is effective are shown below.

```
a~ a
a
% increase the length of the tie
-\tweak minimum-length #5
~ a
```
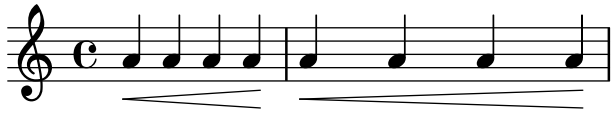


```
a1
\compressFullBarRests
```

```
R1*23
% increase the length of the rest bar
\once \override MultiMeasureRest.minimum-length = #20
R1*23
a1
```
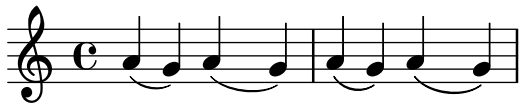



```
a \< a a a \!
% increase the length of the hairpin
\override Hairpin.minimum-length = #20
a \< a a a \!
```



This override can also be used to increase the length of slurs and phrasing slurs:

```
a( g)
a
-\tweak minimum-length #5
( g)

a\( g\)
a
-\tweak minimum-length #5
\( g\)
```
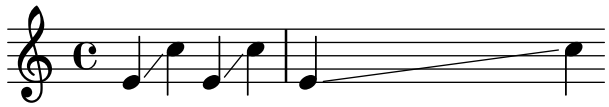


For some layout objects, the `minimum-length` property becomes effective only if the `set-spacing-rods` procedure is called explicitly. To do this, the `springs-and-rods` property should be set to `ly:spanner::set-spacing-rods`. For example, the minimum length of a glissando has no effect unless the `springs-and-rods` property is set:

```
% default
e \glissando c'

% not effective alone
\once \override Glissando.minimum-length = #20
e, \glissando c'
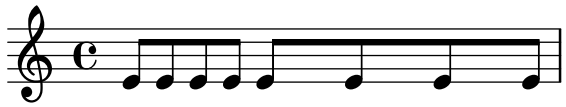
% effective only when both overrides are present
\once \override Glissando.minimum-length = #20
\once \override Glissando.springs-and-rods = #ly:spanner::set-spacing-rods
e, \glissando c'
```



The same is true of the `Beam` object:

```
% not effective alone
\once \override Beam.minimum-length = #20
e8 e e e

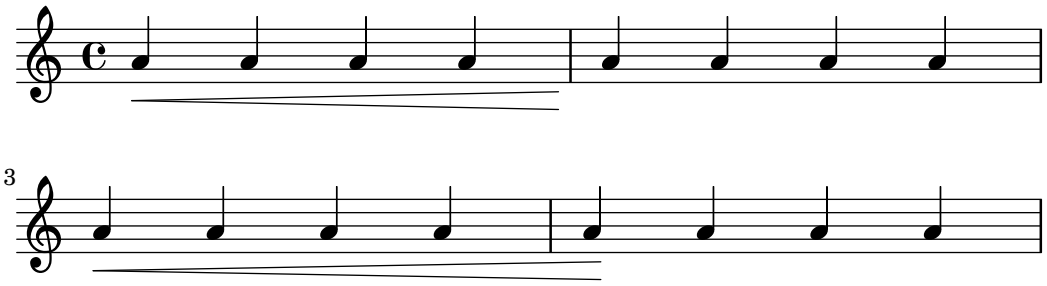
% effective only when both overrides are present
\once \override Beam.minimum-length = #20
\once \override Beam.springs-and-rods = #ly:spanner::set-spacing-rods
e8 e e e
```



The `to-barline` *property*

The second useful property of the `spanner-interface` is `to-barline`. By default this is true, causing hairpins and other spanners which are terminated on the first note of a measure to end instead on the immediately preceding bar line. If set to false, the spanner will extend beyond the bar line and end on the note itself:

```
a \< a a a a \! a a a \break
\override Hairpin.to-barline = ##f
a \< a a a a \! a a a
```



This property is not effective for all spanners. For example, setting it to `#t` has no effect on slurs or phrasing slurs or on other spanners for which terminating on the bar line would not be meaningful.
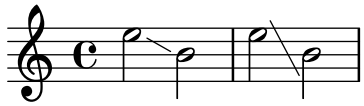
Using the `line-spanner-interface`

Objects which support the `line-spanner-interface` include

- `DynamicTextSpanner`
- `Glissando`
- `TextSpanner`
- `TrillSpanner`
- `VoiceFollower`

The routine responsible for drawing the stencils for these spanners is `ly:line-interface::print`. This routine determines the exact location of the two end points and draws a line between them, in the style requested. The locations of the two end points of the spanner are computed on-the-fly, but it is possible to override their Y-coordinates. The properties which need to be specified are nested two levels down within the property hierarchy, but the syntax of the `\override` command is quite simple:

```
e2 \glissando b
\once \override Glissando.bound-details.left.Y = #3
\once \override Glissando.bound-details.right.Y = #-2
e2 \glissando b
```

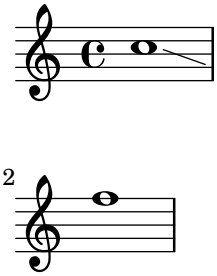


The units for the `Y` property are `staff-space`s, with the center line of the staff being the zero point. For the glissando, this is the value for `Y` at the X-coordinate corresponding to the center point of each note head, if the line is imagined to be extended to there.

If `Y` is not set, the value is computed from the vertical position of the corresponding attachment point of the spanner.

In case of a line break, the values for the end points are specified by the `left-broken` and `right-broken` sub-lists of `bound-details`. For example:

```
\override Glissando.breakable = ##t
\override Glissando.bound-details.right-broken.Y = #-3
c1 \glissando \break
f1
```



A number of further properties of the `left` and `right` sub-lists of the `bound-details` property may be modified in the same way as `Y`:

`Y` This sets the Y-coordinate of the end point, in `staff-space`s offset from the staff center line. By default, it is the center of the bound object, so a glissando points to the vertical center of the note head.

For horizontal spanners, such as text spanners and trill spanners, it is hardcoded to 0.
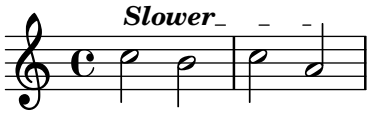
`attach-dir` This determines where the line starts and ends in the X-direction, relative to the bound object. So, a value of `-1` (or `LEFT`) makes the line start/end at the left side of the note head it is attached to.

`X` This is the absolute X-coordinate of the end point. It is usually computed on the fly, and overriding it has little useful effect.

`stencil` Line spanners may have symbols at the beginning or end, which is contained in this sub-property. This is for internal use; it is recommended that `text` be used instead.

`text` This is a markup that is evaluated to yield the stencil. It is used to put *cresc.*, *tr* and other text on horizontal spanners.

```
\override TextSpanner.bound-details.left.text
   = \markup { \small \bold Slower }
c2\startTextSpan b c a\stopTextSpan
```

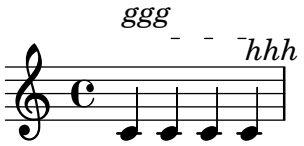


`stencil-align-dir-y`
`stencil-offset`

Without setting one of these, the stencil is simply put at the end-point, centered on the line, as defined by the `X` and `Y` sub-properties. Setting either `stencil-align-dir-y` or `stencil-offset` will move the symbol at the edge vertically relative to the end point of the line:

```
\override TextSpanner.bound-details.left.stencil-align-dir-y = #-2
\override TextSpanner.bound-details.right.stencil-align-dir-y = #UP

\override TextSpanner.bound-details.left.text = #"ggg"
\override TextSpanner.bound-details.right.text = #"hhh"
c4^\startTextSpan c c c \stopTextSpan
```
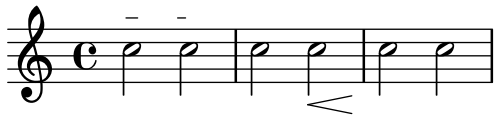


Note that negative values move the text *up*, contrary to the effect that might be expected, as a value of `-1` or `DOWN` means align the *bottom* edge of the text with the spanner line. A value of `1` or `UP` aligns the top edge of the text with the spanner line.

`arrow` Setting this sub-property to `#t` produces an arrowhead at the end of the line.

`padding` This sub-property controls the space between the specified end point of the line and the actual end. Without padding, a glissando would start and end in the center of each note head.

The music function `\endSpanners` terminates the spanner which starts on the immediately following note prematurely. It is terminated after exactly one note, or at the following bar line if `to-barline` is true and a bar line occurs before the next note.

```
\endSpanners
c2 \startTextSpan c2 c2
\endSpanners
c2 \< c2 c2
```



When using `\endSpanners` it is not necessary to close \startTextSpan with \stopTextSpan, nor is it necessary to close hairpins with `\!`.

Vedi anche

Internals Reference: Sezione "TextSpanner" in *Guida al Funzionamento Interno*, Sezione "Glissando" in *Guida al Funzionamento Interno*, Sezione "VoiceFollower" in *Guida al Funzionamento Interno*, Sezione "TrillSpanner" in *Guida al Funzionamento Interno*, Sezione "line-spanner-interface" in *Guida al Funzionamento Interno*.

5.4.6 Visibility of objects

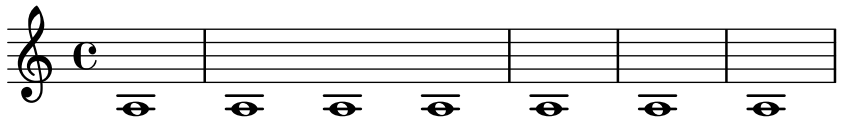
There are four main ways in which the visibility of layout objects can be controlled: their stencil can be removed, they can be made transparent, they can be colored white, or their `break-visibility` property can be overridden. The first three apply to all layout objects; the last to just a few – the *breakable* objects. The Learning Manual introduces these four techniques, see Sezione "Visibility and color of objects" in *Manuale di Apprendimento*.

There are also a few other techniques which are specific to certain layout objects. These are covered under Special considerations.

Removing the stencil

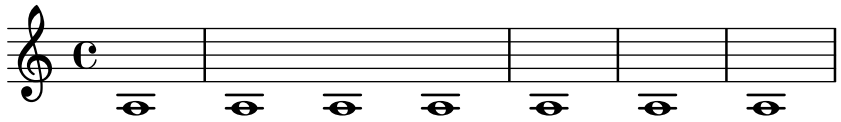
Every layout object has a stencil property. By default this is set to the specific function which draws that object. If this property is overridden to `#f` no function will be called and the object will not be drawn. The default action can be recovered with `\revert`.

```
a1 a
\override Score.BarLine.stencil = ##f
a a
\revert Score.BarLine.stencil
a a a
```



This rather common operation has a shortcut `\omit`:
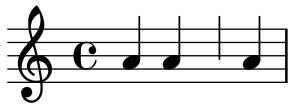
```
a1 a
\omit Score.BarLine
a a
\undo \omit Score.BarLine
a a a
```



Making objects transparent
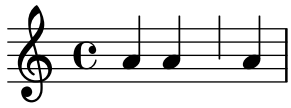
Every layout object has a transparent property which by default is set to `#f`. If set to `#t` the object still occupies space but is made invisible.

```
a4 a
\once \override NoteHead.transparent = ##t
a a
```



This rather common operation has a shortcut `\hide`:

```
a4 a
\once \hide NoteHead
a a
```



Painting objects white

Every layout object has a color property which by default is set to `black`. If this is overridden to `white` the object will be indistinguishable from the white background. However, if the object crosses other objects the color of the crossing points will be determined by the order in which they are drawn, and this may leave a ghostly image of the white object, as shown here:

```
\override Staff.Clef.color = #white
a1
```



This may be avoided by changing the order of printing the objects. All layout objects have a `layer` property which should be set to an integer. Objects with the lowest value of `layer` are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a `layer` value of `1`, although a few objects, including `StaffSymbol` and `BarLine`, are assigned a value of `0`. The order of printing objects with the same value of `layer` is indeterminate.

In the example above the white clef, with a default `layer` value of `1`, is drawn after the staff lines (default `layer` value `0`), so overwriting them. To change this, the `Clef` object must be given in a lower value of `layer`, say `-1`, so that it is drawn earlier:

```
\override Staff.Clef.color = #white
\override Staff.Clef.layer = #-1
a1
```



Using break-visibility

Most layout objects are printed only once, but some like bar lines, clefs, time signatures and key signatures, may need to be printed twice when a line break occurs – once at the end of the line and again at the start of the next line. Such objects are called *breakable*, and have a property, the `break-visibility` property to control their visibility at the three positions in which they may appear – at the start of a line, within a line if they are changed, and at the end of a line if a change takes place there.

For example, the time signature by default will be printed at the start of the first line, but nowhere else unless it changes, when it will be printed at the point at which the change occurs. If this change occurs at the end of a line the new time signature will be printed at the start of the next line and a cautionary time signature will be printed at the end of the previous line as well.

This behaviour is controlled by the `break-visibility` property, which is explained in Sezione "Visibility and color of objects" in *Manuale di Apprendimento*. This property takes a vector of three booleans which, in order, determine whether the object is printed at the end of, within the body of, or at the beginning of a line. Or to be more precise, before a line break, where there is no line break, or after a line break.

Alternatively, these eight combinations may be specified by pre-defined functions, defined in '`scm/output-lib.scm`', where the last three columns indicate whether the layout objects will be visible in the positions shown at the head of the columns:
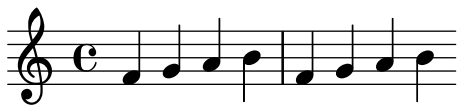
Function form	**Vector form**	**Before break**	**At no break**	**After break**
`all-visible`	`'#(#t #t #t)`	yes	yes	yes
`begin-of-line-visible`	`'#(#f #f #t)`	no	no	yes
`center-visible`	`'#(#f #t #f)`	no	yes	no
`end-of-line-visible`	`'#(#t #f #f)`	yes	no	no
`begin-of-line-invisible`	`'#(#t #t #f)`	yes	yes	no
`center-invisible`	`'#(#t #f #t)`	yes	no	yes
`end-of-line-invisible`	`'#(#f #t #t)`	no	yes	yes
`all-invisible`	`'#(#f #f #f)`	no	no	no
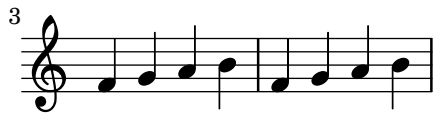
The default settings of `break-visibility` depend on the layout object. The following table shows all the layout objects of interest which are affected by `break-visibility` and the default setting of this property:

Layout object	**Usual context**	**Default setting**
`BarLine`	`Score`	calculated
`BarNumber`	`Score`	`begin-of-line-visible`
`BreathingSign`	`Voice`	`begin-of-line-invisible`
`Clef`	`Staff`	`begin-of-line-visible`
`Custos`	`Staff`	`end-of-line-visible`
`DoublePercentRepeat`	`Voice`	`begin-of-line-invisible`
`KeyCancellation`	`Staff`	`begin-of-line-invisible`
`KeySignature`	`Staff`	`begin-of-line-visible`
`ClefModifier`	`Staff`	`begin-of-line-visible`
`RehearsalMark`	`Score`	`end-of-line-invisible`
`TimeSignature`	`Staff`	`all-visible`

The example below shows the use of the vector form to control the visibility of bar lines:

```
f4 g a b
f4 g a b
% Remove bar line at the end of the current line
\once \override Score.BarLine.break-visibility = ##(#f #t #t)
\break
f4 g a b
f4 g a b
```





Although all three components of the vector used to override `break-visibility` must be present, not all of them are effective with every layout object, and some combinations may even give errors. The following limitations apply:
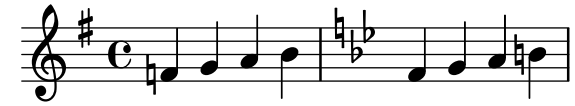
- Bar lines cannot be printed at start of line.
- A bar number cannot be printed at the start of the first line unless it is set to be different from 1.
- Clef – see below
- Double percent repeats are either all printed or all suppressed. Use begin-of line-invisible to print and all-invisible to suppress.
- Key signature – see below
- ClefModifier – see below
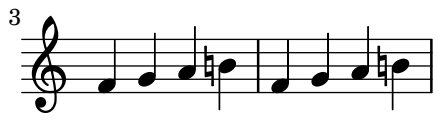
Special considerations

Visibility following explicit changes

The `break-visibility` property controls the visibility of key signatures and changes of clef only at the start of lines, i.e. after a break. It has no effect on the visibility of the key signature or clef following an explicit key change or an explicit clef change within or at the end of a line. In the following example the key signature following the explicit change to B-flat major is still visible, even though `all-invisible` is set.
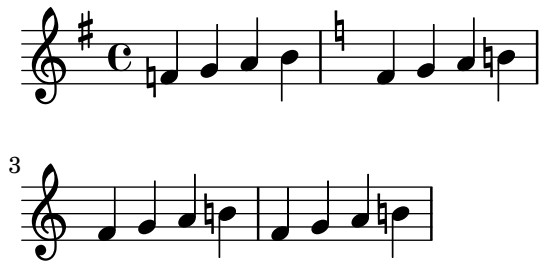
```
\key g \major
f4 g a b
% Try to remove all key signatures
\override Staff.KeySignature.break-visibility = #all-invisible
\key bes \major
f4 g a b
\break
f4 g a b
f4 g a b
```





The visibility of such explicit key signature and clef changes is controlled by the `explicitKeySignatureVisibility` and `explicitClefVisibility` properties. These are the equivalent of the `break-visibility` property and both take a vector of three booleans or the predefined functions listed above, exactly like `break-visibility`. Both are properties of the Staff context, not the layout objects themselves, and so they are set using the `\set` command. Both are set by default to `all-visible`. These properties control only the visibility of key signatures and clefs resulting from explicit changes and do not affect key signatures and clefs at the beginning of lines; `break-visibility` must still be overridden in the appropriate object to remove these.
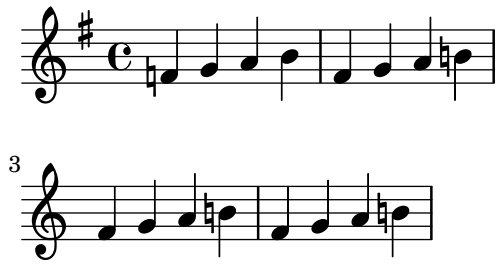
```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Staff.KeySignature.break-visibility = #all-invisible
\key bes \major
f4 g a b \break
f4 g a b
f4 g a b
```



Visibility of cancelling accidentals

To remove the cancelling accidentals printed at an explicit key change, set the Staff context property `printKeyCancellation` to `#f`:

```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.printKeyCancellation = ##f
\override Staff.KeySignature.break-visibility = #all-invisible
\key bes \major
f4 g a b \break
f4 g a b
f4 g a b
```
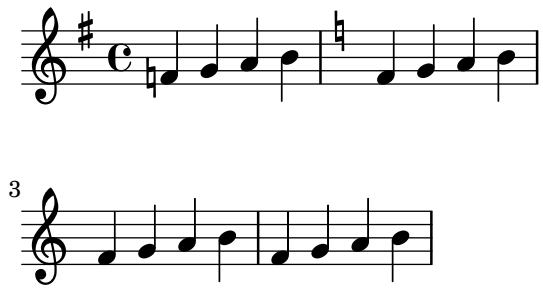


With these overrides only the accidentals before the notes remain to indicate the change of key.

Note that when changing the key to C major or A minor the cancelling accidentals would be the *only* indication of the key change. In this case setting `printKeyCancellation` to `#f` has no effect:
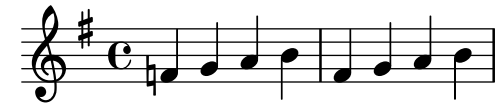
```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.printKeyCancellation = ##f
\key c \major
f4 g a b \break
f4 g a b
```
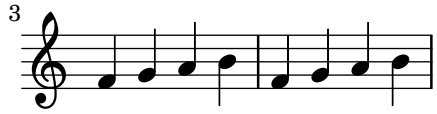
```
f4 g a b
```



To suppress the cancelling accidentals even when the key is changed to C major or A minor, override the visibility of the `KeyCancellation` grob instead:

```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Staff.KeyCancellation.break-visibility = #all-invisible
\key c \major
f4 g a b \break
f4 g a b
f4 g a b
```





Automatic bars

As a special case, the printing of bar lines can also be turned off by setting the `automaticBars` property in the Score context. If set to `#f`, bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` predefined command, measures are still counted. Bar generation will resume according to that count if this property is later set to `#t`. When set to `#f`, line breaks can occur only at explicit `\bar` commands.

Transposed clefs

The small transposition symbol on transposed clefs is produced by the `ClefModifier` layout object. Its visibility is automatically inherited from the `Clef` object, so it is not necessary to apply any required `break-visibility` overrides to the `ClefModifier` layout objects to suppress transposition symbols for invisible clefs.

For explicit clef changes, the `explicitClefVisibility` property controls both the clef symbol and any transposition symbol associated with it.

Vedi anche

Learning Manual: Sezione “Visibility and color of objects” in *Manuale di Apprendimento*.

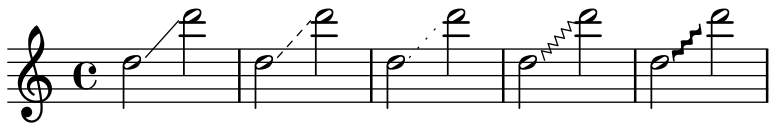
5.4.7 Line styles

Some performance indications, e.g., *rallentando* and *accelerando* and *trills* are written as text and are extended over many measures with lines, sometimes dotted or wavy.

These all use the same routines as the glissando for drawing the texts and the lines, and tuning their behavior is therefore also done in the same way. It is done with a spanner, and the routine responsible for drawing the spanners is `ly:line-interface::print`. This routine determines the exact location of the two *span points* and draws a line between them, in the style requested.
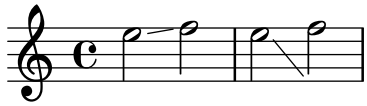
Here is an example showing the different line styles available, and how to tune them.

```
d2 \glissando d'2
\once \override Glissando.style = #'dashed-line
d,2 \glissando d'2
\override Glissando.style = #'dotted-line
d,2 \glissando d'2
\override Glissando.style = #'zigzag
d,2 \glissando d'2
\override Glissando.style = #'trill
d,2 \glissando d'2
```



The locations of the end-points of the spanner are computed on-the-fly for every graphic object, but it is possible to override these:

```
e2 \glissando f
\once \override Glissando.bound-details.right.Y = #-2
e2 \glissando f
```



The value for `Y` is set to `-2` for the right end point. The left side may be similarly adjusted by specifying `left` instead of `right`.

If `Y` is not set, the value is computed from the vertical position of the left and right attachment points of the spanner.

Other adjustments of spanners are possible, for details, see Sezione 5.4.5 [Spanners], pagina 587.

5.4.8 Rotating objects

Both layout objects and elements of markup text can be rotated by any angle about any point, but the method of doing so differs.
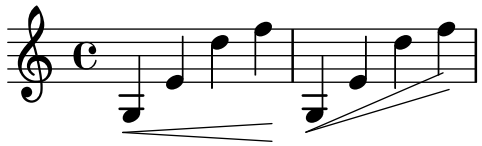
Rotating layout objects

All layout objects which support the `grob-interface` can be rotated by setting their `rotation` property. This takes a list of three items: the angle of rotation counter-clockwise, and the x and y coordinates of the point relative to the object's reference point about which the rotation is to be performed. The angle of rotation is specified in degrees and the coordinates in staff-spaces.

The angle of rotation and the coordinates of the rotation point must be determined by trial and error.

There are only a few situations where the rotation of layout objects is useful; the following example shows one situation where they may be:
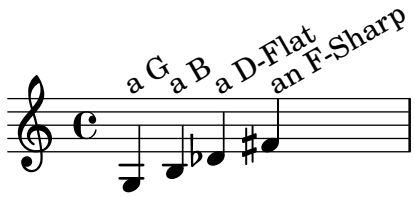
```
g4\< e' d' f\!
\override Hairpin.rotation = #'(20 -1 0)
g,,4\< e' d' f\!
```



Rotating markup

All markup text can be rotated to lie at any angle by prefixing it with the `\rotate` command. The command takes two arguments: the angle of rotation in degrees counter-clockwise and the text to be rotated. The extents of the text are not rotated: they take their values from the extremes of the x and y coordinates of the rotated text. In the following example the `outside-staff-priority` property for text is set to `#f` to disable the automatic collision avoidance, which would push some of the text too high.

```
\override TextScript.outside-staff-priority = ##f
g4^\markup { \rotate #30 "a G" }
b^\markup { \rotate #30 "a B" }
des^\markup { \rotate #30 "a D-Flat" }
fis^\markup { \rotate #30 "an F-Sharp" }
```



5.5 Advanced tweaks

This section discusses various approaches to fine tuning the appearance of the printed score.

Vedi anche

Learning Manual: Sezione “Tweaking output” in *Manuale di Apprendimento*, Sezione “Other sources of information” in *Manuale di Apprendimento*.

Notation Reference: Sezione 5.2 [Explaining the Internals Reference], pagina 572, Sezione 5.3 [Modifying properties], pagina 575.

Extending LilyPond: Sezione “Interfaces for programmers” in *Estendere*.

Installed Files: ‘`scm/define-grobs.scm`’.

Snippets: Sezione “Tweaks and overrides” in *Frammenti di codice*.

Internals Reference: Sezione “All layout objects” in *Guida al Funzionamento Interno*.

5.5.1 Aligning objects

Graphical objects which support the `self-alignment-interface` and/or the `side-position-interface` can be aligned to a previously placed object in a variety of ways. For a list of these objects, see Sezione "self-alignment-interface" in *Guida al Funzionamento Interno* and Sezione "side-position-interface" in *Guida al Funzionamento Interno*.

All graphical objects have a reference point, a horizontal extent and a vertical extent. The horizontal extent is a pair of numbers giving the displacements from the reference point of the left and right edges, displacements to the left being negative. The vertical extent is a pair of numbers giving the displacement from the reference point to the bottom and top edges, displacements down being negative.

An object's position on a staff is given by the values of the `X-offset` and `Y-offset` properties. The value of `X-offset` gives the displacement from the X coordinate of the reference point of the parent object, and the value of `Y-offset` gives the displacement from the center line of the staff. The values of `X-offset` and `Y-offset` may be set directly or may be set to be calculated by procedures in order to achieve alignment with the parent object.

Nota: Many objects have special positioning considerations which cause any setting of `X-offset` or `Y-offset` to be ignored or modified, even though the object supports the `self-alignment-interface`. Overriding the `X-offset` or `Y-offset` properties to a fixed value causes the respective `self-alignment` property to be disregarded.

For example, an accidental can be repositioned vertically by setting `Y-offset` but any changes to `X-offset` have no effect.

Rehearsal marks may be aligned with breakable objects such as bar lines, clef symbols, time signature symbols and key signatures. There are special properties to be found in the `break-aligned-interface` for positioning rehearsal marks on such objects.

Vedi anche
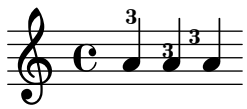
Notation Reference: [Using the break-alignable-interface], pagina 602.

Extending LilyPond: Sezione "Callback functions" in *Estendere*.

Setting `X-offset` and `Y-offset` directly

Numerical values may be given to the `X-offset` and `Y-offset` properties of many objects. The following example shows three notes with the default fingering position and the positions with `X-offset` and `Y-offset` modified.

```
a-3
a
-\tweak X-offset #0
-\tweak Y-offset #0
-3
a
-\tweak X-offset #-1
-\tweak Y-offset #1
-3
```



Using the side-position-interface

An object which supports the side-position-interface can be placed next to its parent object so that the specified edges of the two objects touch. The object may be placed above, below, to the right or to the left of the parent. The parent cannot be specified; it is determined by the order of elements in the input stream. Most objects have the associated note head as their parent.

The values of the side-axis and direction properties determine where the object is to be placed, as follows:

side-axis property	direction property	Placement
0	-1	left
0	1	right
1	-1	below
1	1	above

When side-axis is 0, X-offset should be set to the procedure ly:side-position-interface::x-aligned-side. This procedure will return the correct value of X-offset to place the object to the left or right side of the parent according to value of direction.

When side-axis is 1, Y-offset should be set to the procedure ly:side-position-interface::y-aligned-side. This procedure will return the correct value of Y-offset to place the object to the top or bottom of the parent according to value of direction.

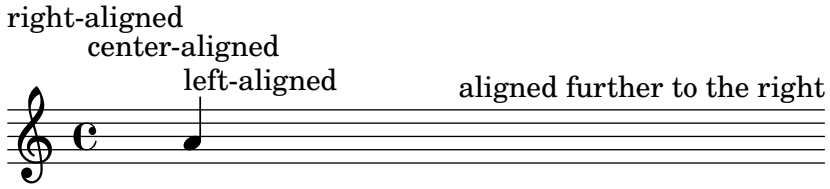
Using the self-alignment-interface

Self-aligning objects horizontally

The horizontal alignment of an object which supports the self-alignment-interface is controlled by the value of the self-alignment-X property, provided the object's X-offset property is set to ly:self-alignment-interface::x-aligned-on-self. self-alignment-X may be given any real value, in units of half the total X extent of the object. Negative values move the object to the right, positive to the left. A value of 0 centers the object on the reference point of its parent, a value of -1 aligns the left edge of the object on the reference point of its parent, and a value of 1 aligns the right edge of the object on the reference point of its parent. The symbols LEFT, CENTER, and RIGHT may be used instead of the values -1, 0, and 1, respectively.

Normally the \override command would be used to modify the value of self-alignment-X, but the \tweak command can be used to separately align several annotations on a single note:

```
a'
-\tweak self-alignment-X #-1
^"left-aligned"
-\tweak self-alignment-X #0
^"center-aligned"
-\tweak self-alignment-X #RIGHT
^"right-aligned"
-\tweak self-alignment-X #-2.5
^"aligned further to the right"
```



Self-aligning objects vertically
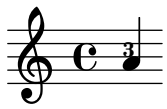
Objects may be aligned vertically in an analogous way to aligning them horizontally if the `Y-offset` property is set to `ly:self-alignment-interface::y-aligned-on-self`. However, other mechanisms are often involved in vertical alignment: the value of `Y-offset` is just one variable taken into account. This may make adjusting the value of some objects tricky. The units are just half the vertical extent of the object, which is usually quite small, so quite large numbers may be required. A value of `-1` aligns the lower edge of the object with the reference point of the parent object, a value of `0` aligns the center of the object with the reference point of the parent, and a value of `1` aligns the top edge of the object with the reference point of the parent. The symbols `DOWN`, `CENTER`, and `UP` may be substituted for `-1`, `0`, and `1`, respectively.

Self-aligning objects in both directions

By setting both `X-offset` and `Y-offset`, an object may be aligned in both directions simultaneously.

The following example shows how to adjust a fingering mark so that it nestles close to the note head.

```
a
-\tweak self-alignment-X #0.5  % move horizontally left
-\tweak Y-offset #ly:self-alignment-interface::y-aligned-on-self
-\tweak self-alignment-Y #-1  % move vertically up
-3  % third finger
```
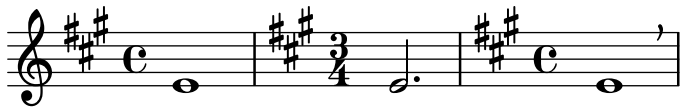


Using the `break-alignable-interface`

Rehearsal marks and bar numbers may be aligned with notation objects other than bar lines. These objects include `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature`, and `time-signature`.

Each type of object has its own default reference point, to which rehearsal marks are aligned:
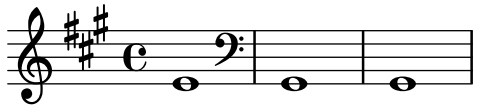
```
% The rehearsal mark will be aligned to the right edge of the Clef
\override Score.RehearsalMark.break-align-symbols = #'(clef)
\key a \major
\clef treble
\mark ""
e1
% The rehearsal mark will be aligned to the left edge of the Time Signature
\override Score.RehearsalMark.break-align-symbols = #'(time-signature)
\key a \major
\clef treble
\time 3/4
\mark ""
e2.
% The rehearsal mark will be centered above the Breath Mark
\override Score.RehearsalMark.break-align-symbols = #'(breathing-sign)
```

```
\key a \major
\clef treble
\time 4/4
e1
\breathe
\mark ""
```



A list of possible target alignment objects may be specified. If some of the objects are invisible at that point due to the setting of `break-visibility` or the explicit visibility settings for keys and clefs, the rehearsal mark or bar number is aligned to the first object in the list which is visible. If no objects in the list are visible the object is aligned to the bar line. If the bar line is invisible the object is aligned to the place where the bar line would be.
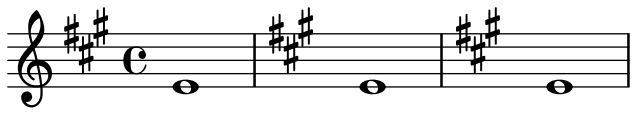
```
% The rehearsal mark will be aligned to the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark ""
e1
% The rehearsal mark will be aligned to the right edge of the Clef
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef bass
\mark ""
gis,,1
% The rehearsal mark will be centered above the Bar Line
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark ""
e''1
```



The alignment of the rehearsal mark relative to the notation object can be changed, as shown in the following example. In a score with multiple staves, this setting should be done for all the staves.
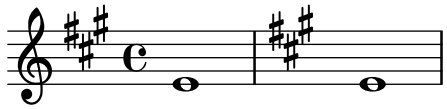
```
% The RehearsalMark will be aligned with the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\key a \major
\clef treble
\time 4/4
```

```
\mark ""
e1
% The RehearsalMark will be centered above the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment = #CENTER
\mark ""
\key a \major
e1
% The RehearsalMark will be aligned with the left edge of the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment = #LEFT
\key a \major
\mark ""
e1
```



The rehearsal mark can also be offset to the right or left of the left edge by an arbitrary amount. The units are staff-spaces:

```
% The RehearsalMark will be aligned with the left edge of the Key Signature
% and then shifted right by 3.5 staff-spaces
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = #3.5
\key a \major
\mark ""
e1
% The RehearsalMark will be aligned with the left edge of the Key Signature
% and then shifted left by 2 staff-spaces
\once \override Score.KeySignature.break-align-anchor = #-2
\key a \major
\mark ""
e1
```



5.5.2 Vertical grouping of grobs

The VerticalAlignment and VerticalAxisGroup grobs work together. VerticalAxisGroup groups together different grobs like Staff, Lyrics, etc. VerticalAlignment then vertically aligns the different grobs grouped together by VerticalAxisGroup. There is usually only one VerticalAlignment per score but every Staff, Lyrics, etc. has its own VerticalAxisGroup.

5.5.3 Modifying stencils

All layout objects have a `stencil` property which is part of the `grob-interface`. By default, this property is usually set to a function specific to the object that is tailor-made to render the symbol which represents it in the output. For example, the standard setting for the `stencil` property of the `MultiMeasureRest` object is `ly:multi-measure-rest::print`.

The standard symbol for any object can be replaced by modifying the `stencil` property to reference a different, specially-written, procedure. This requires a high level of knowledge of the

internal workings of LilyPond, but there is an easier way which can often produce adequate results.

This is to set the `stencil` property to the procedure which prints text – `ly:text-interface::print` – and to add a `text` property to the object which is set to contain the markup text which produces the required symbol. Due to the flexibility of markup, much can be achieved – see in particular ⟨undefined⟩ [Graphic notation inside markup], pagina ⟨undefined⟩.

The following example demonstrates this by changing the note head symbol to a cross within a circle.

```
XinO = {
  \once \override NoteHead.stencil = #ly:text-interface::print
  \once \override NoteHead.text = \markup {
    \combine
      \halign #-0.7 \draw-circle #0.85 #0.2 ##f
      \musicglyph #"noteheads.s2cross"
  }
}
\relative c'' {
  a a \XinO a a
}
```



Any of the glyphs in the feta Font can be supplied to the `\musicglyph` markup command – see Sezione A.8 [The Feta font], pagina 632.

Vedi anche

Notation Reference: ⟨undefined⟩ [Graphic notation inside markup], pagina ⟨undefined⟩, ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩, Sezione A.10 [Text markup commands], pagina 653, Sezione A.8 [The Feta font], pagina 632.

5.5.4 Modifying shapes

Modifying ties and slurs

`Ties`, `Slurs`, `PhrasingSlurs`, `LaissezVibrerTies` and `RepeatTies` are all drawn as third-order Bézier curves. If the shape of the tie or slur which is calculated automatically is not optimum, the shape may be modified manually in two ways:

a. by specifying the displacements to be made to the control points of the automatically calculated Bézier curve, or

b. by explicitly specifying the positions of the four control points required to define the wanted curve.

Both methods are explained below. The first method is more suitable if only slight adjustments to the curve are required; the second may be better for creating curves which are related to just a single note.

Cubic Bézier curves
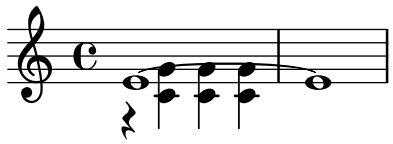
Third-order or cubic Bézier curves are defined by four control points. The first and fourth control points are precisely the starting and ending points of the curve. The intermediate two control

points define the shape. Animations showing how the curve is drawn can be found on the web, but the following description may be helpful. The curve starts from the first control point heading directly towards the second, gradually bending over to head towards the third and continuing to bend over to head towards the fourth, arriving there travelling directly from the third control point. The curve is entirely contained in the quadrilateral defined by the four control points. Translations, rotations and scaling of the control points all result in exactly the same operations on the curve.

Specifying displacements from current control points

In this example the automatic placement of the tie is not optimum, and `\tieDown` would not help.

```
<<
  { e1~ e }
\\
  { r4 <g c,> <g c,> <g c,> }
>>
```



Adjusting the control points of the tie with `\shape` allows the collisions to be avoided.

The syntax of `\shape` is

`[-]\shape` *`displacements item`*
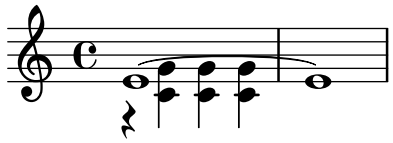
This will reposition the control-points of *item* by the amounts given by *displacements*. The *displacements* argument is a list of number pairs or a list of such lists. Each element of a pair represents the displacement of one of the coordinates of a control-point. If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

In other words, the `\shape` function can act as either a `\once\override` command or a `\tweak` command depending on whether the *item* argument is a grob name, like "Slur", or a music expression, like "(". The *displacements* argument specifies the displacements of the four control points as a list of four pairs of (dx . dy) values in units of staff-spaces (or a list of such lists if the curve has more than one segment).

The leading hyphen is required if and only if the `\tweak` form is being used.

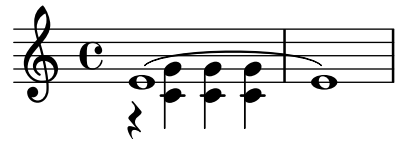
So, using the same example as above and the `\once\override` form of `\shape`, this will raise the tie by half a staff-space:

```
<<
  {
    \shape #'((0 . 0.5) (0 . 0.5) (0 . 0.5) (0 . 0.5)) Tie
    e1~ e
  }
\\
  { r4 <g c,> <g c,> <g c,> }
>>
```
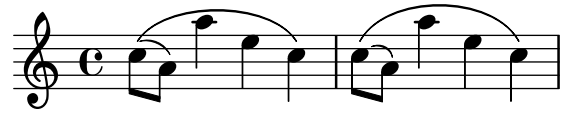


This positioning of the tie is better, but maybe it should be raised more in the center. The following example does this, this time using the alternative `\tweak` form:

```
<<
  {
    e1-\shape #'((0 . 0.5) (0 . 1) (0 . 1) (0 . 0.5)) ~ e
  }
\\
  { r4 <g c,> <g c,> <g c,> }
>>
```
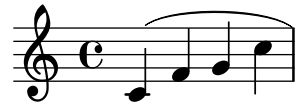


Changes to the horizontal positions of the control points may be made in the same way, and two different curves starting at the same musical moment may also be shaped:
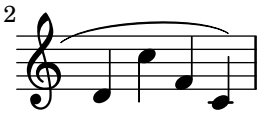
```
c8(\( a) a'4 e c\)
\shape #'((0.7 . -0.4) (0.5 . -0.4) (0.3 . -0.3) (0 . -0.2)) Slur
\shape #'((0 . 0) (0 . 0.5) (0 . 0.5) (0 . 0)) PhrasingSlur
c8(\( a) a'4 e c\)
```



The `\shape` function can also displace the control points of curves which stretch across line breaks. Each piece of the broken curve can be given its own list of offsets. If changes to a particular segment are not needed, the empty list can serve as a placeholder. In this example the line break makes the single slur look like two:

```
c4( f g c
\break
d,4 c' f, c)
```
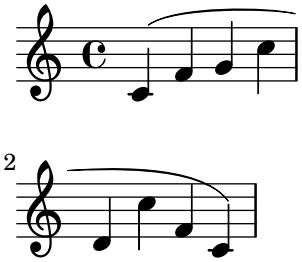




Changing the shapes of the two halves of the slur makes it clearer that the slur continues over the line break:
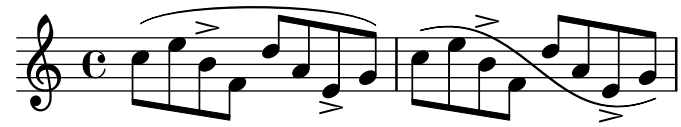
```
% () may be used as a shorthand for ((0 . 0) (0 . 0) (0 . 0) (0 . 0))
% if any of the segments does not need to be changed
\shape #'(
           (( 0 . 0) (0 . 0) (0 . 0) (0 . 1))
           ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
         ) Slur
c4( f g c
```

```
\break
d,4 c' f, c)
```



If an S-shaped curve is required the control points must always be adjusted manually — LilyPond will never select such shapes automatically.

```
c8( e b-> f d' a e-> g)
\shape #'((0 . -1) (5.5 . -0.5) (-5.5 . -10.5) (0 . -5.5)) PhrasingSlur
c8\( e b-> f d' a e-> g\)
```

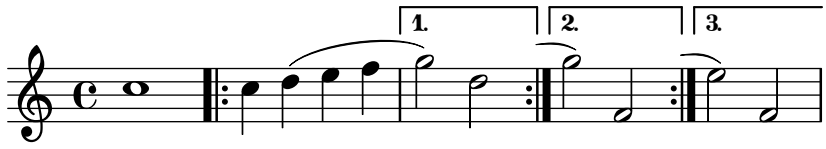


Specifying control points explicitly

The coordinates of the Bézier control points are specified in units of staff-spaces. The X coordinate is relative to the reference point of the note to which the tie or slur is attached, and the Y coordinate is relative to the staff center line. The coordinates are specified as a list of four pairs of decimal numbers (reals). One approach is to estimate the coordinates of the two end points, and then guess the two intermediate points. The optimum values are then found by trial and error. Be aware that these values may need to be manually adjusted if any further changes are made to the music or the layout.

One situation where specifying the control points explicitly is preferable to specifying displacements is when they need to be specified relative to a single note. Here is an example of this. It shows one way of indicating a slur extending into alternative sections of a volta repeat.

```
c1
\repeat volta 3 { c4 d( e f }
\alternative {
  { g2) d }
  {
    g2
    % create a slur and move it to a new position
    % the <> is just an empty chord to carry the slur termination
    -\tweak control-points #'((-2 . 3.8) (-1 . 3.9) (0 . 4) (1 . 3.4)) ( <> )
    f,
  }
  {
    e'2
    % create a slur and move it to a new position
    -\tweak control-points #'((-2 . 3) (-1 . 3.1) (0 . 3.2) (1 . 2.4)) ( <> )
    f,
  }
}
```



Problemi noti e avvertimenti

It is not possible to modify shapes of ties or slurs by changing the `control-points` property if there are multiple ties or slurs at the same musical moment – the `\tweak` command will also not work in this case. However, the `tie-configuration` property of `TieColumn` can be overridden to set start line and direction as required.

Vedi anche
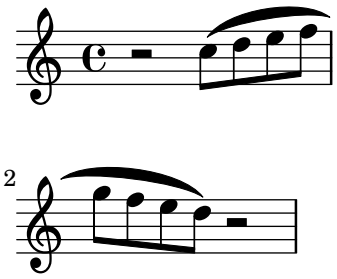
Internals Reference: Sezione “TieColumn” in *Guida al Funzionamento Interno*.

5.5.5 Modifying broken spanners

Using `\alterBroken`

When a spanner crosses a line break or breaks, each piece inherits the attributes of the original spanner. Thus, ordinary tweaking of a broken spanner applies the same modifications to each of its segments. In the example below, overriding `thickness` affects the slur on either side of the line break.

```
r2
\once\override Slur.thickness = 10
c8( d e f
\break
g8 f e d) r2
```



Independently modifying the appearance of individual pieces of a broken spanner is possible with the `\alterBroken` command. This command can produce either an `\override` or a `\tweak` of a spanner property.

The syntax for `\alterBroken` is

`[-]\alterBroken` *`property values item`*
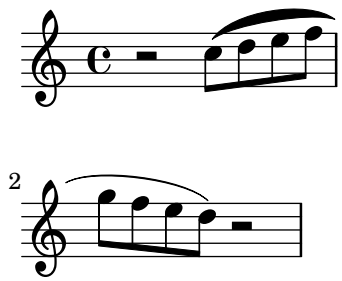
The argument *values* is a list of values, one for each broken piece. If *item* is a grob name like `Slur` or `Staff.PianoPedalBracket`, the result is an `\override` of the specified grob type. If *item* is a music expression such as “(” or “[” the result is the same music expression with an appropriate tweak applied.

The leading hyphen must be used with the `\tweak` form. Do not add it when `\alterBroken` is used as an `\override`.

In its `\override` usage, `\alterBroken` may be prefaced by `\once` or `\temporary` and reverted by using `\revert` with *property*.

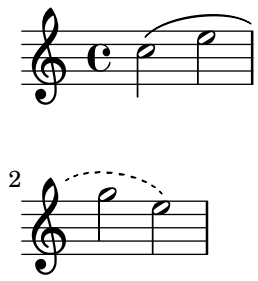
The following code applies an independent `\override` to each of the slur segments in the previous example:

```
r2
\alterBroken thickness #'(10 1) Slur
c8( d e f
\break
g8 f e d) r2
```



The `\alterBroken` command may be used with any spanner object, including `Tie`, `PhrasingSlur`, `Beam` and `TextSpanner`. For example, an editor preparing a scholarly edition may wish to indicate the absence of part of a phrasing slur in a source by dashing only the segment which has been added. The following example illustrates how this can be done, in this case using the `\tweak` form of the command:

```
% The empty list is conveniently used below, because it is the
% default setting of dash-definition, resulting in a solid curve.
c2-\alterBroken dash-definition #'(() ((0 1.0 0.4 0.75))) \(e
\break
g2 e\)
```



It is important to understand that `\alterBroken` will set each piece of a broken spanner to the corresponding value in *values*. When there are fewer values than pieces, any additional piece will be assigned the empty list. This may lead to undesired results if the layout property is not set to the empty list by default. In such cases, each segment should be assigned an appropriate value.

Problemi noti e avvertimenti

Line breaks may occur in different places following changes in layout. Settings chosen for `\alterBroken` may be unsuitable for a spanner that is no longer broken or is split into more segments than before. Explicit use of `\break` can guard against this situation.

The `\alterBroken` command is ineffective for spanner properties accessed before line-breaking such as `direction`.

Vedi anche

Extending LilyPond: Sezione “Difficult tweaks” in *Estendere*.

5.5.6 Unpure-pure containers

Unpure-pure containers are useful for overriding *Y-axis* spacing calculations - specifically `Y-offset` and `Y-extent` - with a Scheme function instead of a literal (i.e. a number or pair).

For certain grobs, the `Y-extent` is based on the `stencil` property, overriding the stencil property of one of these will require an additional `Y-extent` override with an unpure-pure container. When a function overrides a `Y-offset` and/or `Y-extent` it is assumed that this will trigger line breaking calculations too early during compilation. So the function is not evaluated at all (usually returning a value of '`0`' or ''`(0 . 0)`') which can result in collisions. A 'pure' function will not affect properties, objects or grob suicides and therefore will always have its Y-axis-related evaluated correctly.

Currently, there are about thirty functions that are already considered 'pure' and Unpure-pure containers are a way to set functions not on this list as 'pure'. The 'pure' function is evaluated *before* any line-breaking and so the horizontal spacing can be adjusted 'in time'. The 'unpure' function is then evaluated *after* line breaking.

Nota: As it is difficult to always know which functions are on this list we recommend that any 'pure' functions you create do not use `Beam` or `VerticalAlignment` grobs.

An unpure-pure container is constructed as follows;

`(ly:make-unpure-pure-container f0 f1)`

where `f0` is a function taking *n* arguments (*n >= 1*) and the first argument must always be the grob. This is the function that gives the actual result. *f1* is the function being labeled as 'pure' that takes *n + 2* arguments. Again, the first argument must always still be the grob but the second and third are 'start' and 'end' arguments.

start and *end* are, for all intents and purposes, dummy values that only matter for `Spanners` (i.e `Hairpin` or `Beam`), that can return different height estimations based on a starting and ending column.

The rest are the other arguments to the first function (which may be none if *n = 1*).
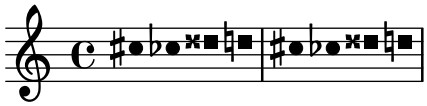
The results of the second function are used as an approximation of the value needed which is then used by the first function to get the real value which is then used for fine-tuning much later during the spacing process.

```
#(define (square-line-circle-space grob)
(let* ((pitch (ly:event-property (ly:grob-property grob 'cause) 'pitch))
       (notename (ly:pitch-notename pitch)))
 (if (= 0 (modulo notename 2))
     (make-circle-stencil 0.5 0.0 #t)
     (make-filled-box-stencil '(0 . 1.0)
                              '(-0.5 . 0.5)))))

squareLineCircleSpace = {
  \override NoteHead.stencil = #square-line-circle-space
}

smartSquareLineCircleSpace = {
  \squareLineCircleSpace
  \override NoteHead.Y-extent =
   #(ly:make-unpure-pure-container
      ly:grob::stencil-height
      (lambda (grob start end) (ly:grob::stencil-height grob)))
}
```

```
\new Voice \with { \remove "Stem_engraver" }
\relative c'' {
  \squareLineCircleSpace
  cis4 ces disis d
  \smartSquareLineCircleSpace
  cis4 ces disis d
}
```



In the first measure, without the unpure-pure container, the spacing engine does not know the width of the note head and lets it collide with the accidentals. In the second measure, with unpure-pure containers, the spacing engine knows the width of the note heads and avoids the collision by lengthening the line accordingly.

Usually for simple calculations nearly-identical functions for both the 'unpure' and 'pure' parts can be used, by only changing the number of arguments passed to, and the scope of, the function.

Nota: If a function is labeled as 'pure' and it turns out not to be, the results can be unexpected.

5.6 Using music functions

Where tweaks need to be reused with different music expressions, it is often convenient to make the tweak part of a *music function*. In this section, we discuss only *substitution* functions, where the object is to substitute a variable into a piece of LilyPond input code. Other more complex functions are described in Sezione "Music functions" in *Estendere*.

5.6.1 Substitution function syntax

Making a function that substitutes a variable into LilyPond code is easy. The general form of these functions is

```
function =
#(define-music-function
     (parser location arg1 arg2 ...)
     (type1? type2? ...)
   #{
     ...music...
   #})
```

where

argN	*n*th argument
typeN?	a scheme *type predicate* for which *argN* must return `#t`.
...music...	normal LilyPond input, using `$` (in places where only Lilypond constructs are allowed) or `#` (to use it as a Scheme value or music function argument or music inside of music lists) to reference arguments (eg. '`#arg1`').

The `parser` and `location` arguments are mandatory, and are used in some advanced situations as described in the 'Extending' manual (see Sezione "Music functions" in *Estendere*). For substitution functions, just be sure to include them.

The list of type predicates is also required. Some of the most common type predicates used in music functions are:

```
boolean?
cheap-list?  (use instead of 'list?'
 for faster processing)
ly:duration?
ly:music?
ly:pitch?
markup?
number?
pair?
string?
symbol?
```

For a list of available type predicates, see Sezione A.20 [Predefined type predicates], pagina 751. User-defined type predicates are also allowed.

Vedi anche

Notation Reference: Sezione A.20 [Predefined type predicates], pagina 751.

Extending Lilypond: Sezione "Music functions" in *Estendere*.

Installed Files: '`lily/music-scheme.cc`', '`scm/c++.scm`', '`scm/lily.scm`'.
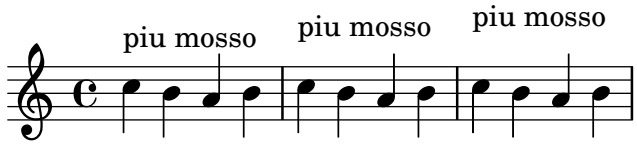
5.6.2 Substitution function examples

This section introduces some substitution function examples. These are not intended to be exhaustive, but rather to demonstrate some of the possibilities of simple substitution functions.

In the first example, a function is defined that simplifies setting the padding of a TextScript:

```
padText =
#(define-music-function
     (parser location padding)
     (number?)
   #{
     \once \override TextScript.padding = #padding
   #})
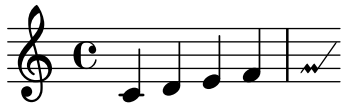
\relative c'' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" b a b
  \padText #2.6
  c4^"piu mosso" b a b
}
```



In addition to numbers, we can use music expressions such as notes for arguments to music functions:

```
custosNote =
#(define-music-function
     (parser location note)
     (ly:music?)
   #{
     \tweak NoteHead.stencil #ly:text-interface::print
     \tweak NoteHead.text
        \markup \musicglyph #"custodes.mensural.u0"
     \tweak Stem.stencil ##f
     #note
   #})

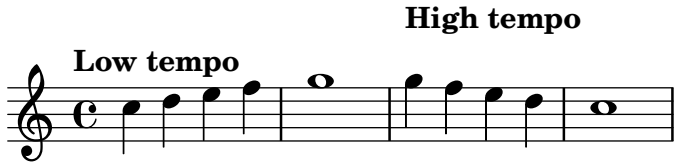
\relative c' { c4 d e f \custosNote g }
```



Substitution functions with multiple arguments can be defined:

```
tempoPadded =
#(define-music-function
     (parser location padding tempotext)
     (number? markup?)
   #{
     \once \override Score.MetronomeMark.padding = #padding
     \tempo \markup { \bold #tempotext }
   #})

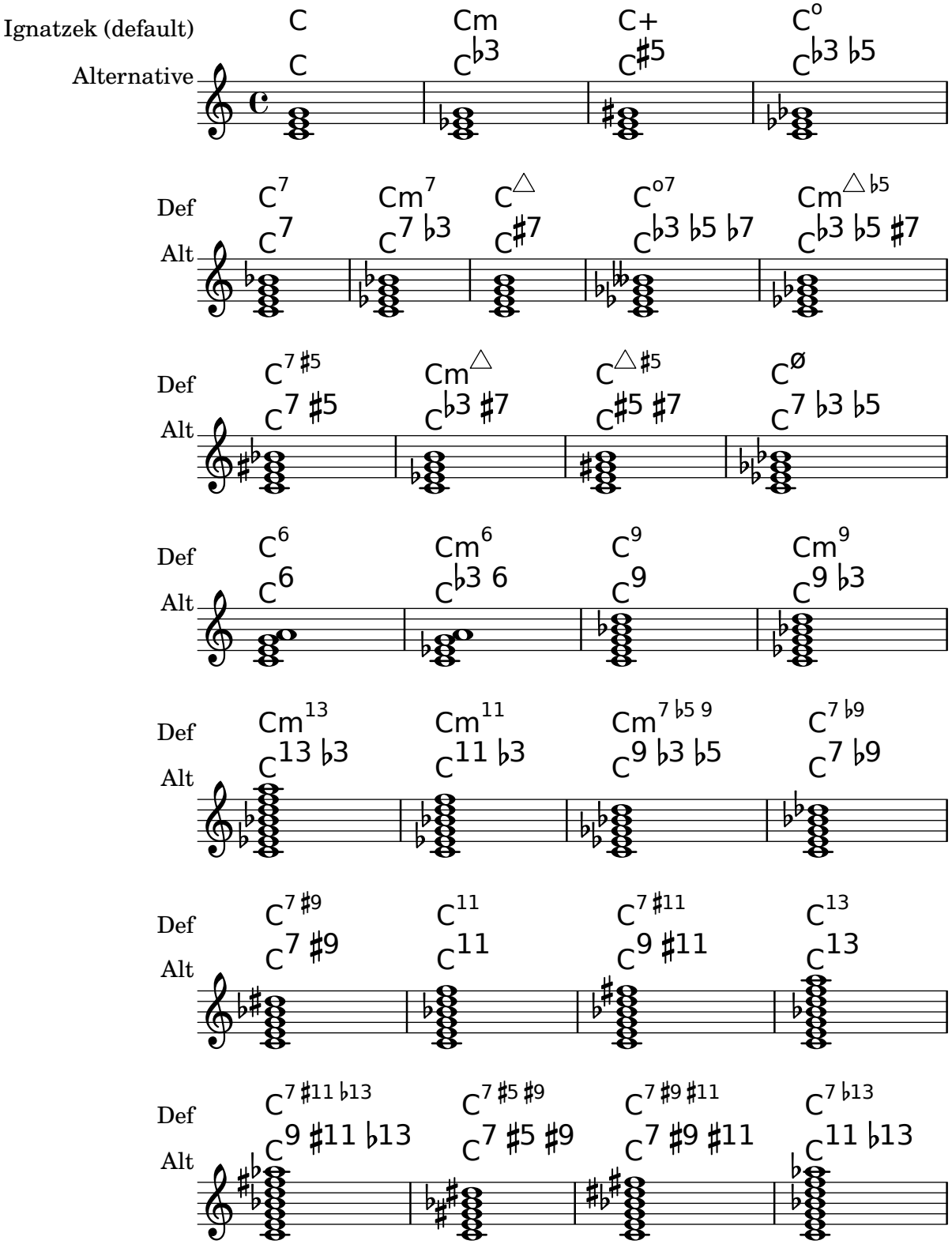
\relative c'' {
  \tempo \markup { "Low tempo" }
  c4 d e f g1
  \tempoPadded #4.0 "High tempo"
  g4 f e d c1
}
```

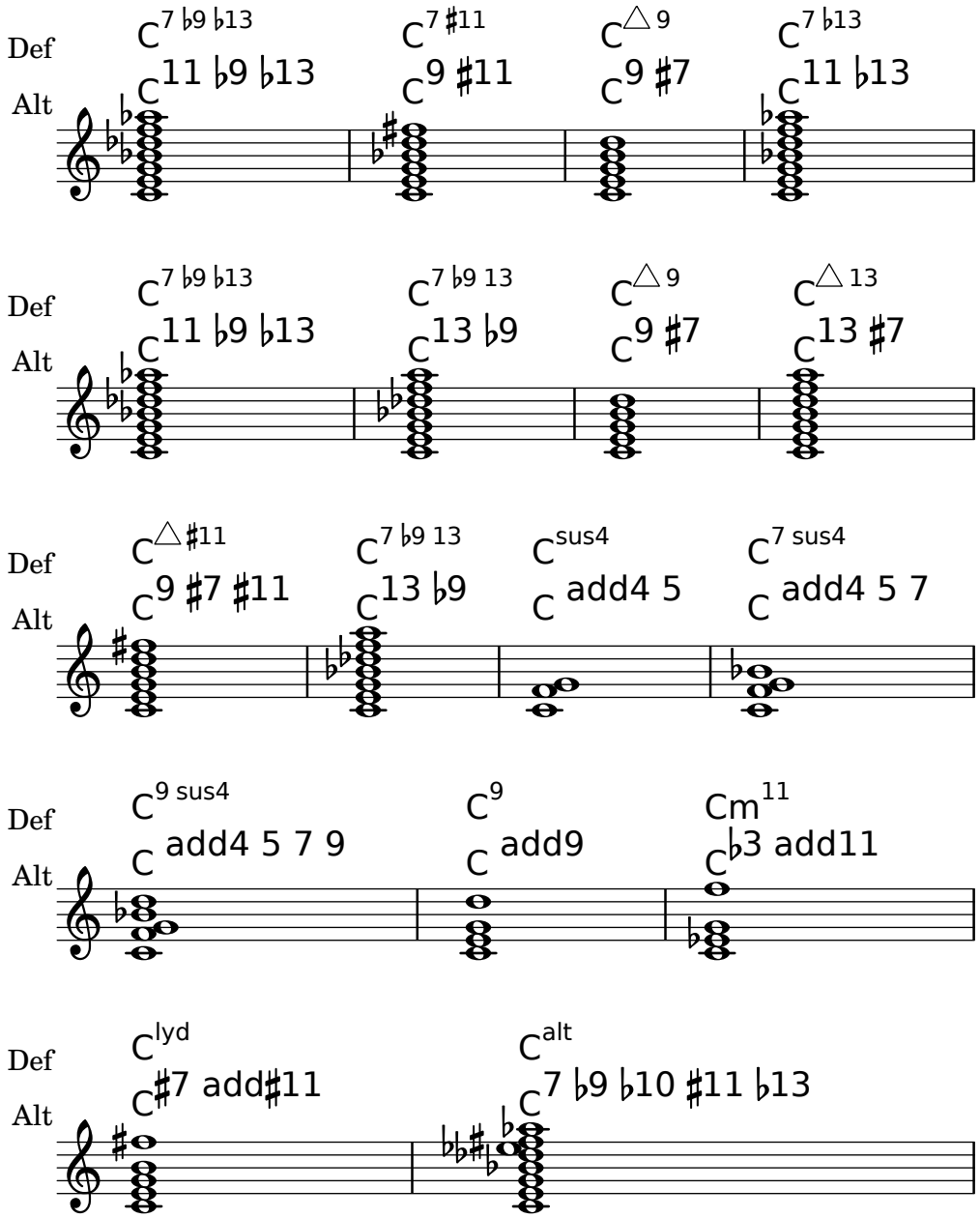


Appendice A Notation manual tables

A.1 Chord name chart
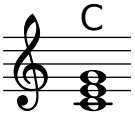
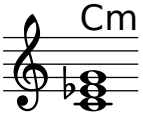
The following chart shows two standard systems for printing chord names, along with the pitches they represent.
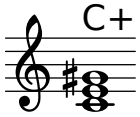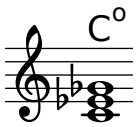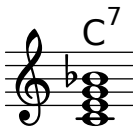
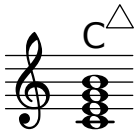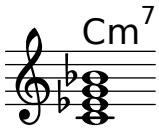
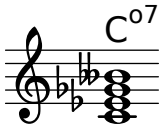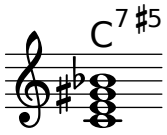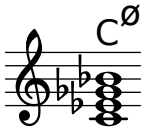
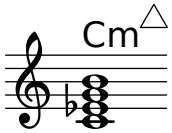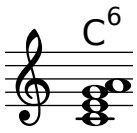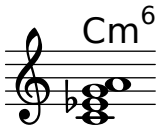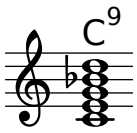


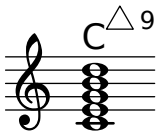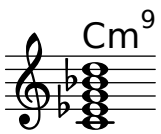
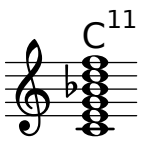
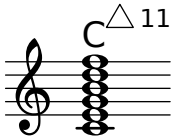
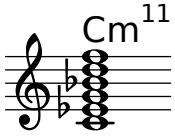


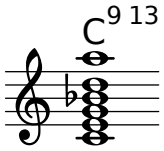
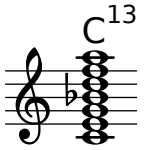
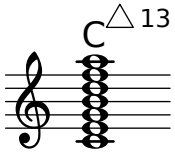
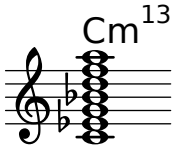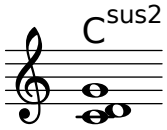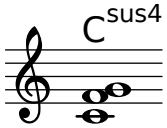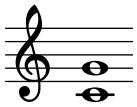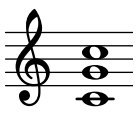
A.2 Common chord modifiers

The following table shows chord modifiers that can be used to generate standard chord structures.

Type	Interval	Modifier	Example	Output
Major	Major third, perfect fifth	`5` or nothing	`c1:5`	
Minor	Minor third, perfect fifth	`m` or `m5`	`c1:m`	

Augmented	Major third, augmented fifth	`aug`	`c1:aug`	
Diminished	Minor third, diminished fifth	`dim`	`c1:dim`	
Dominant seventh	Major triad, minor seventh	`7`	`c1:7`	
Major seventh	Major triad, major seventh	`maj7` or `maj`	`c1:maj7`	
Minor seventh	Minor triad, minor seventh	`m7`	`c1:m7`	
Diminished seventh	Diminished triad, diminished seventh	`dim7`	`c1:dim7`	
Augmented seventh	Augmented triad, minor seventh	`aug7`	`c1:aug`	
Half-diminished seventh	Diminished triad, minor seventh	`m7.5-`	`c1:m7.5-`	
Minor-major seventh	Minor triad, major seventh	`m7+`	`m7+`	

Major sixth	Major triad, sixth	`6`	`c1:6`	
Minor sixth	Minor triad, sixth	`m6`	`c1:m6`	
Dominant ninth	Dominant seventh, major ninth	`9`	`c1:9`	
Major ninth	Major seventh, major ninth	`maj9`	`c1:maj9`	
Minor ninth	Minor seventh, major ninth	`m9`	`c1:m9`	
Dominant eleventh	Dominant ninth, perfect eleventh	`11`	`c1:11`	
Major eleventh	Major ninth, perfect eleventh	`maj11`	`c1:maj11`	
Minor eleventh	Minor ninth, perfect eleventh	`m11`	`c1:m11`	

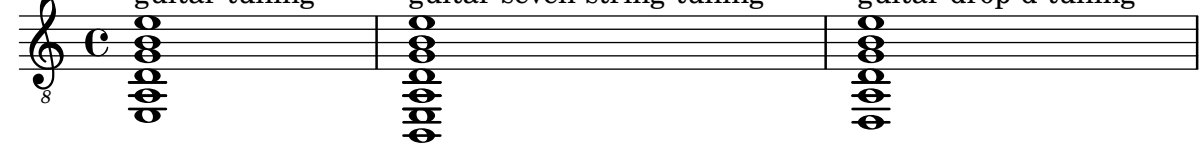
Dominant thirteenth	Dominant ninth, major thirteenth	`13`	`c1:13`	
Dominant thirteenth	Dominant eleventh, major thirteenth	`13.11`	`c1:13.11`	
Major thirteenth	Major eleventh, major thirteenth	`maj13.11`	`c1:maj13.11`	
Minor thirteenth	Minor eleventh, major thirteenth	`m13.11`	`c1:m13.11`	
Suspended second	Major second, perfect fifth	`sus2`	`c1:sus2`	
Suspended fourth	Perfect fourth, perfect fifth	`sus4`	`c1:sus4`	
Power chord (two-voiced)	Perfect fifth	`1.5`	`\powerChords c1:1.5`	
Power chord (three-voiced)	Perfect fifth, octave	`1.5.8`	`\powerChords c1:1.5.8`	

A.3 Predefined string tunings

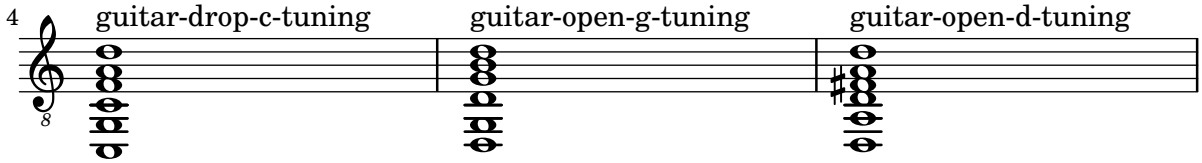
The chart below shows the predefined string tunings.

Guitar tunings

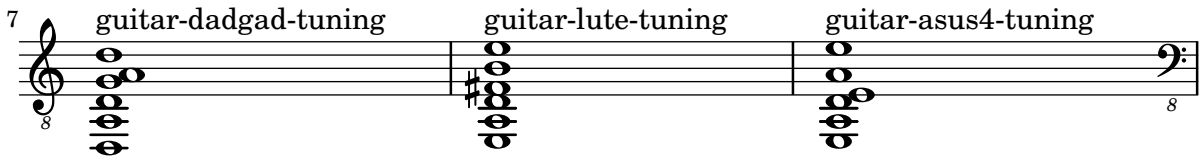
guitar-tuning guitar-seven-string-tuning guitar-drop-d-tuning



guitar-drop-c-tuning guitar-open-g-tuning guitar-open-d-tuning

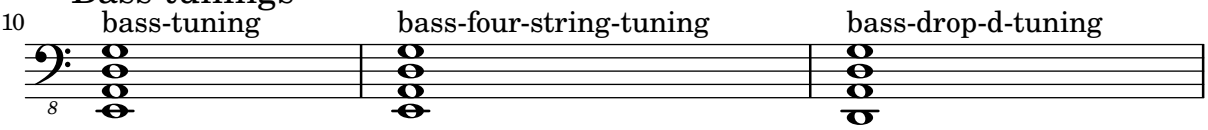


guitar-dadgad-tuning guitar-lute-tuning guitar-asus4-tuning

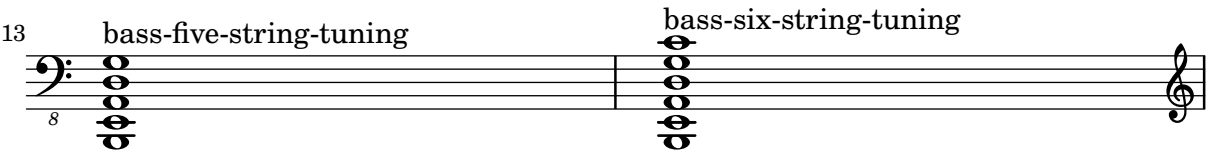


Bass tunings

bass-tuning bass-four-string-tuning bass-drop-d-tuning

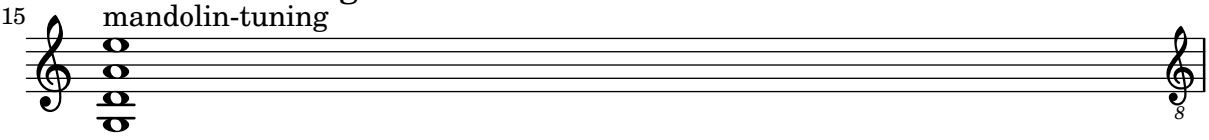


bass-five-string-tuning bass-six-string-tuning
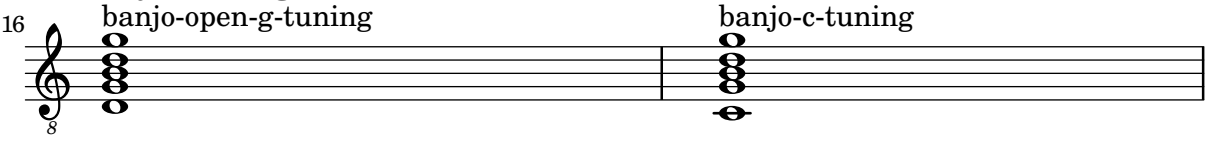


Mandolin tunings

mandolin-tuning

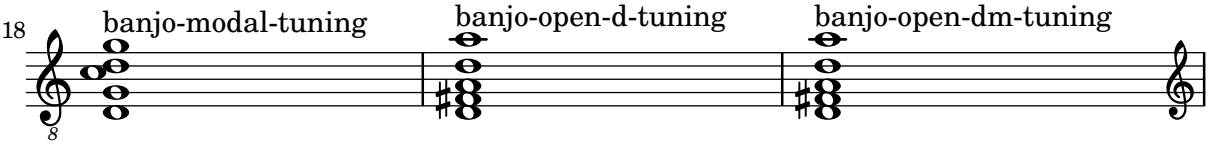


Banjo tunings

banjo-open-g-tuning banjo-c-tuning

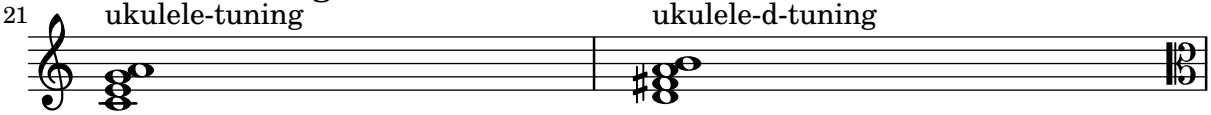


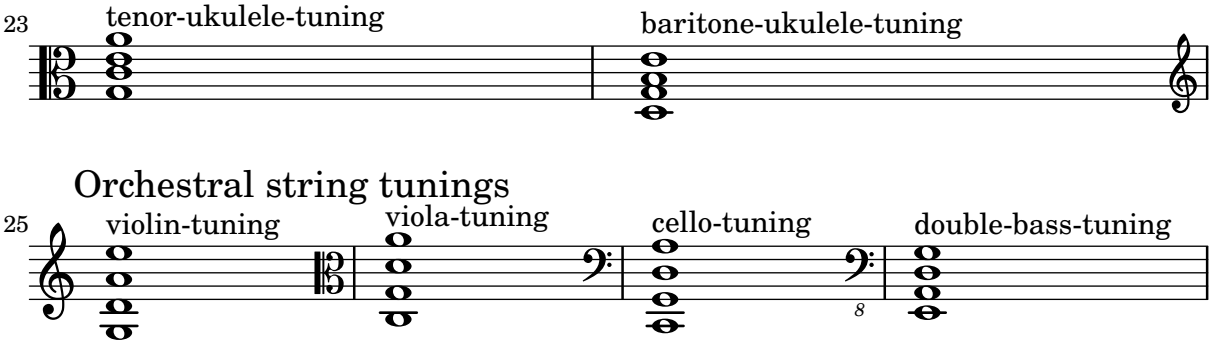
banjo-modal-tuning banjo-open-d-tuning banjo-open-dm-tuning



Ukulele tunings

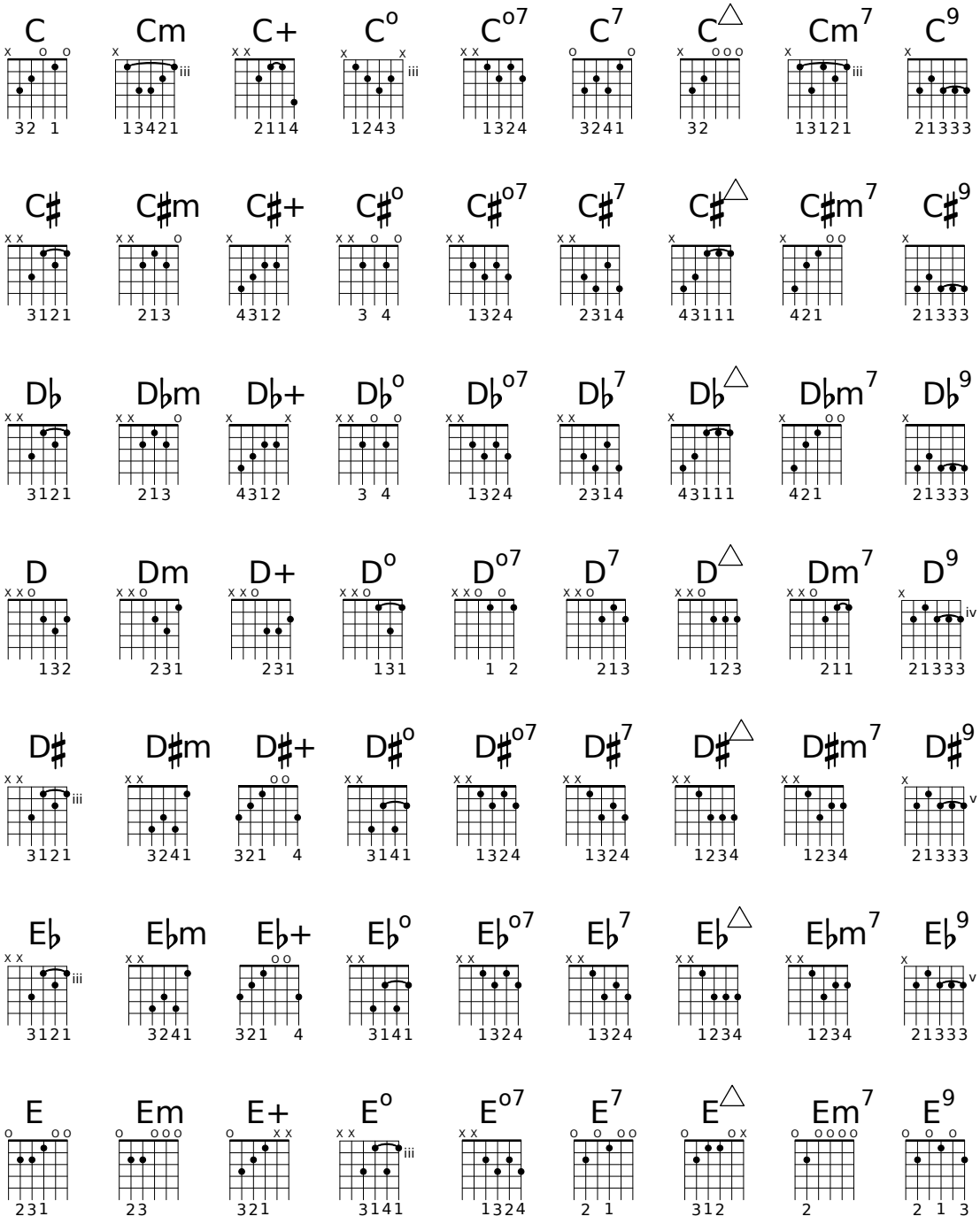
ukulele-tuning ukulele-d-tuning

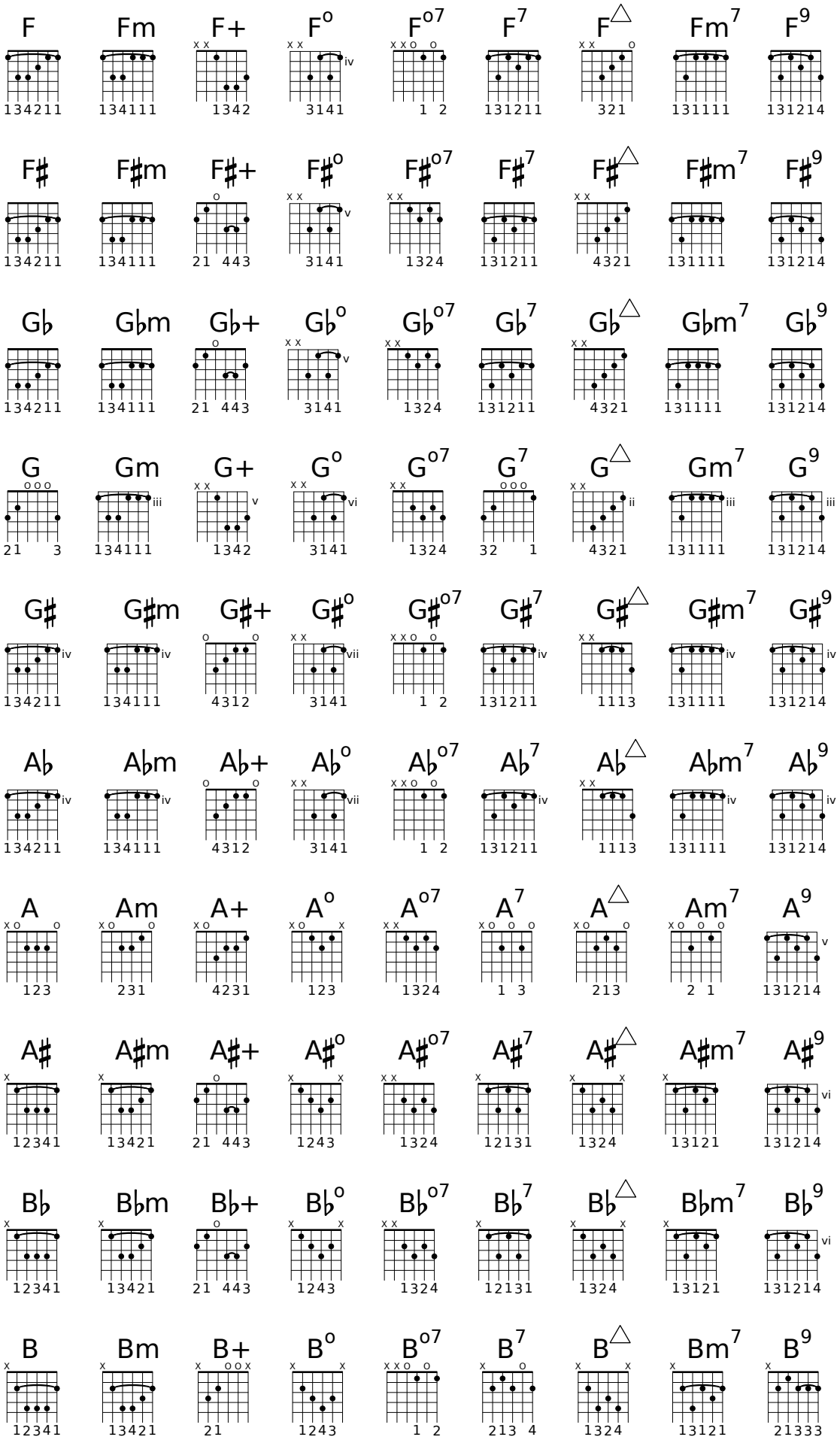




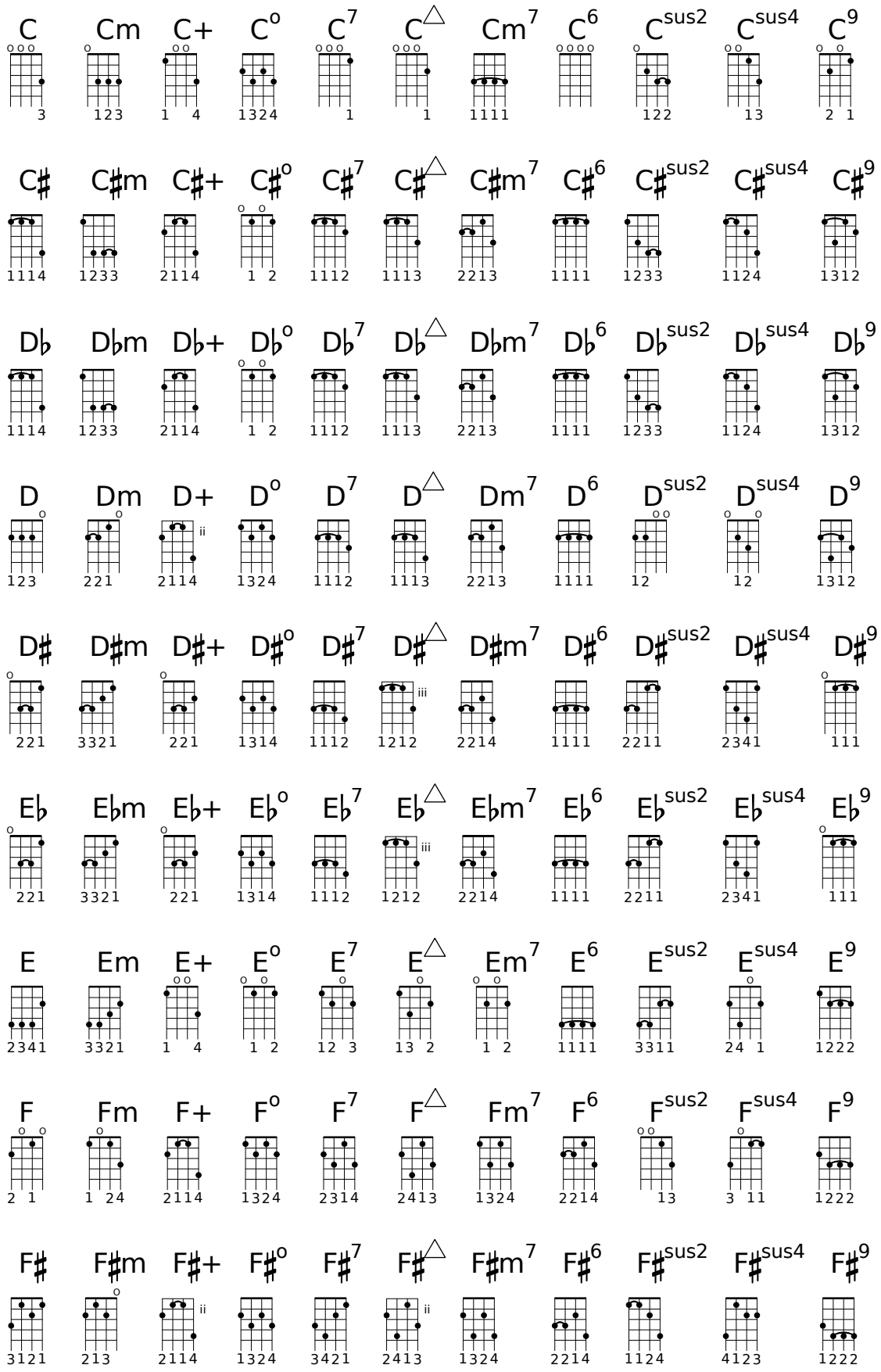
A.4 Predefined fretboard diagrams

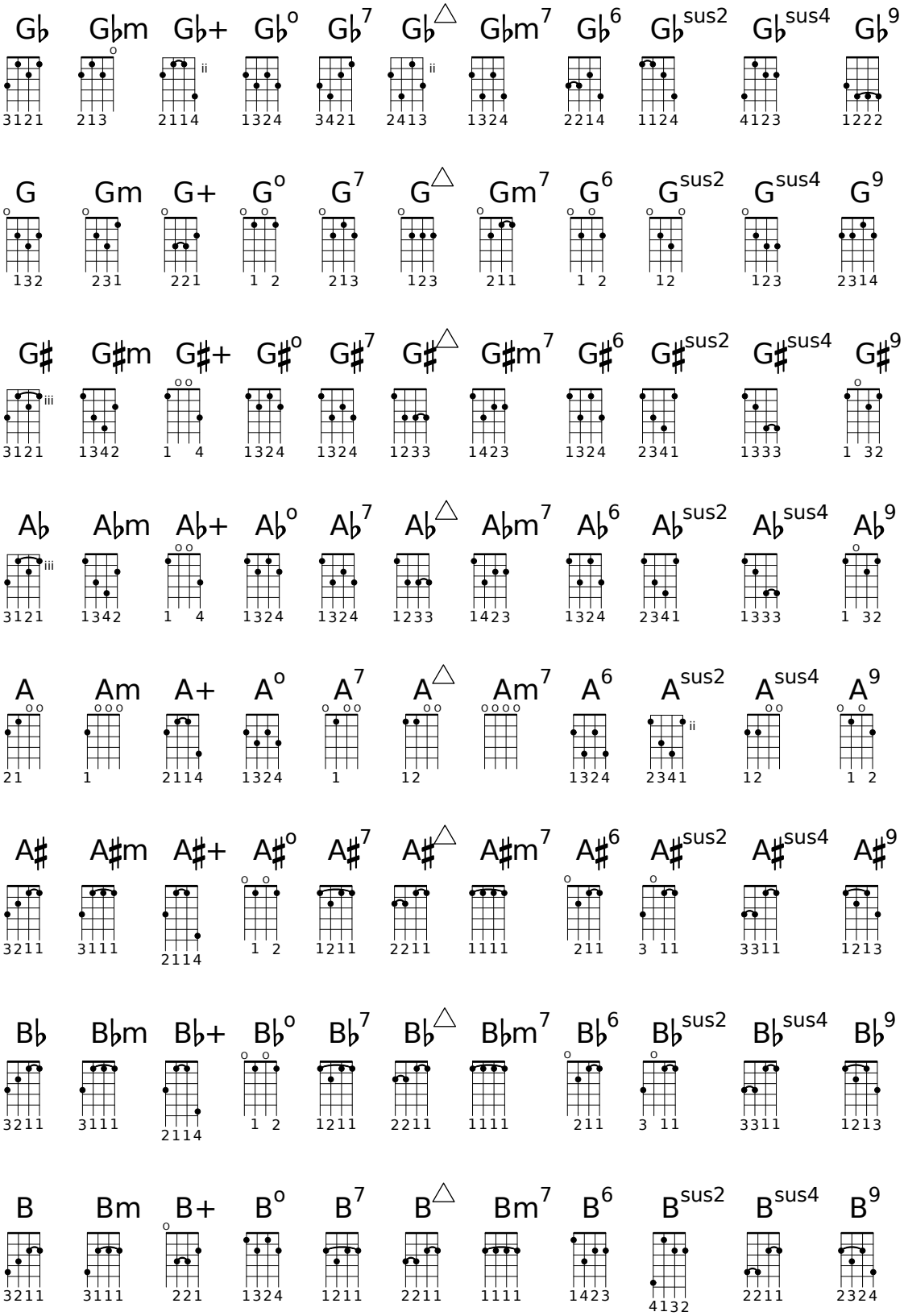
Diagrams for Guitar



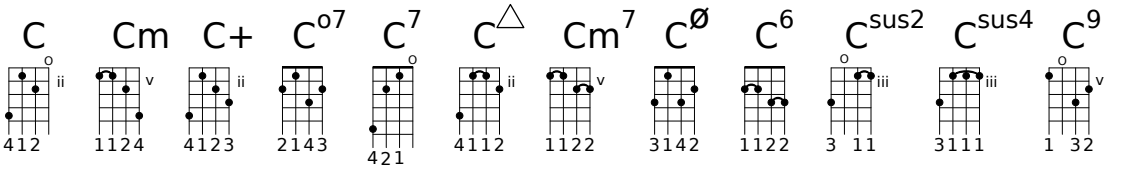

F Fm F+ F° F°7 F7 F△ Fm7 F9
F♯ F♯m F♯+ F♯° F♯°7 F♯7 F♯△ F♯m7 F♯9
G♭ G♭m G♭+ G♭° G♭°7 G♭7 G♭△ G♭m7 G♭9
G Gm G+ G° G°7 G7 G△ Gm7 G9
G♯ G♯m G♯+ G♯° G♯°7 G♯7 G♯△ G♯m7 G♯9
A♭ A♭m A♭+ A♭° A♭°7 A♭7 A♭△ A♭m7 A♭9
A Am A+ A° A°7 A7 A△ Am7 A9
A♯ A♯m A♯+ A♯° A♯°7 A♯7 A♯△ A♯m7 A♯9
B♭ B♭m B♭+ B♭° B♭°7 B♭7 B♭△ B♭m7 B♭9
B Bm B+ B° B°7 B7 B△ Bm7 B9

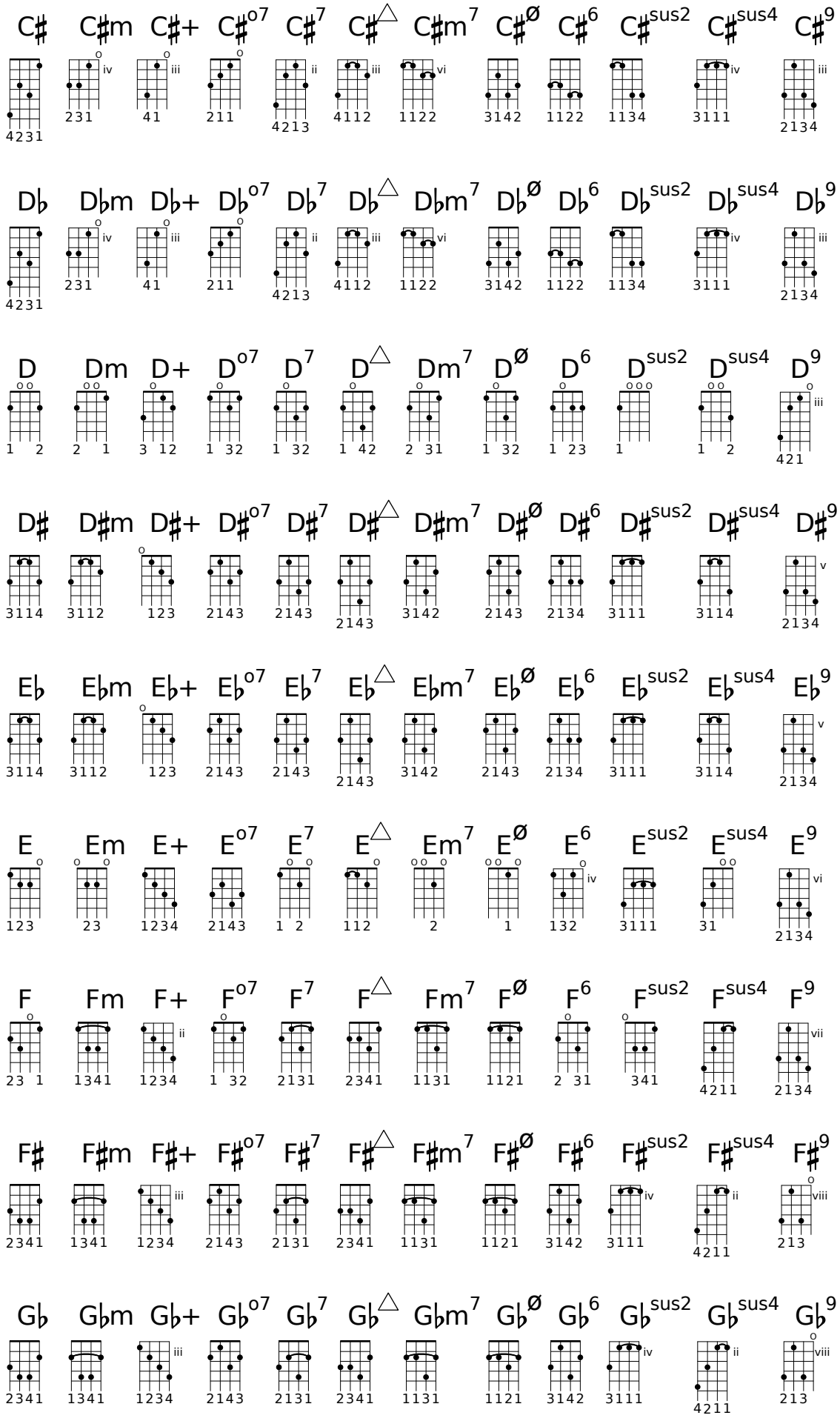
Diagrams for Ukulele

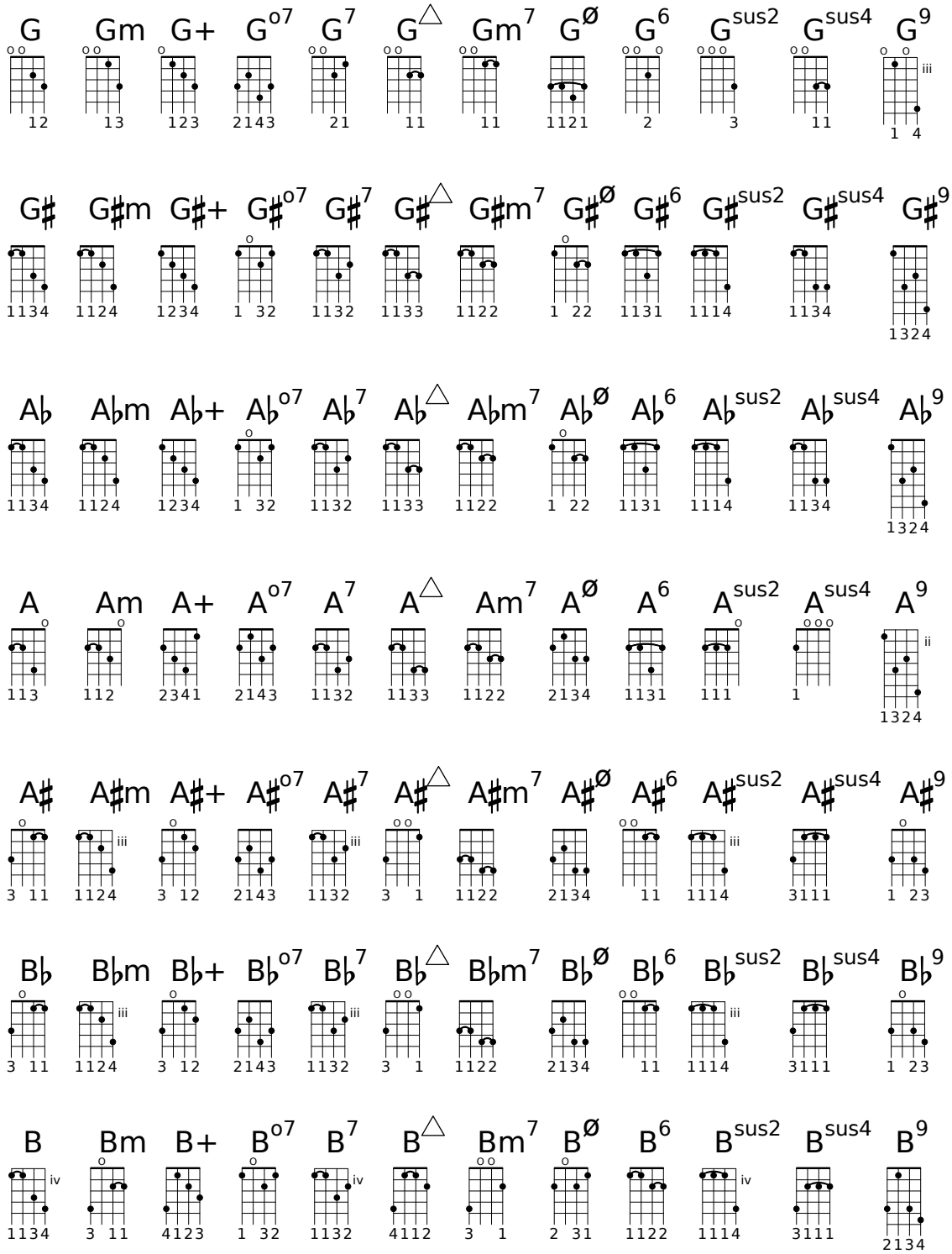




Diagrams for Mandolin




C♯ C♯m C♯+ C♯o7 C♯7 C♯△ C♯m7 C♯ø C♯6 C♯sus2 C♯sus4 C♯9
4231 231 41 211 4213 4112 1122 3142 1122 1134 3111 2134
D♭ D♭m D♭+ D♭o7 D♭7 D♭△ D♭m7 D♭ø D♭6 D♭sus2 D♭sus4 D♭9
4231 231 41 211 4213 4112 1122 3142 1122 1134 3111 2134
D Dm D+ Do7 D7 D△ Dm7 Dø D6 Dsus2 Dsus4 D9
1 2 2 1 3 12 1 32 1 32 1 42 2 31 1 32 1 23 1 1 2 421
D♯ D♯m D♯+ D♯o7 D♯7 D♯△ D♯m7 D♯ø D♯6 D♯sus2 D♯sus4 D♯9
3114 3112 123 2143 2143 2143 3142 2143 2134 3111 3114 2134
E♭ E♭m E♭+ E♭o7 E♭7 E♭△ E♭m7 E♭ø E♭6 E♭sus2 E♭sus4 E♭9
3114 3112 123 2143 2143 2143 3142 2143 2134 3111 3114 2134
E Em E+ Eo7 E7 E△ Em7 Eø E6 Esus2 Esus4 E9
123 23 1234 2143 1 2 112 2 1 132 3111 31 2134
F Fm F+ Fo7 F7 F△ Fm7 Fø F6 Fsus2 Fsus4 F9
23 1 1341 1234 1 32 2131 2341 1131 1121 2 31 341 4211 2134
F♯ F♯m F♯+ F♯o7 F♯7 F♯△ F♯m7 F♯ø F♯6 F♯sus2 F♯sus4 F♯9
2341 1341 1234 2143 2131 2341 1131 1121 3142 3111 4211 213
G♭ G♭m G♭+ G♭o7 G♭7 G♭△ G♭m7 G♭ø G♭6 G♭sus2 G♭sus4 G♭9
2341 1341 1234 2143 2131 2341 1131 1121 3142 3111 4211 213



A.5 Predefined paper sizes

Paper sizes are defined in '`scm/paper.scm`'

The "ISO 216" A Series

`"a10"` (26 x 37 mm)

`"a9"` (37 x 52 mm)

`"a8"` (52 x 74 mm)

`"a7"` (74 x 105 mm)

`"a6"` (105 x 148 mm)

`"a5"` (148 x 210 mm)

`"a4"` (210 x 297 mm)

`"a3"` (297 x 420 mm)

`"a2"` (420 x 594 mm)

`"a1"` (594 x 841 mm)

`"a0"` (841 x 1189 mm)

The "ISO 216" B Series

`"b10"` (31 x 44 mm)

`"b9"` (44 x 62 mm)

`"b8"` (62 x 88 mm)

`"b7"` (88 x 125 mm)

`"b6"` (125 x 176 mm)

`"b5"` (176 x 250 mm)

`"b4"` (250 x 353 mm)

`"b3"` (353 x 500 mm)

`"b2"` (500 x 707 mm)

`"b1"` (707 x 1000 mm)

`"b0"` (1000 x 1414 mm)

Two extended sizes as defined in "DIN 476"

`"4a0"` (1682 x 2378 mm)

`"2a0"` (1189 x 1682 mm)

"ISO 269" standard C series

`"c10"` (28 x 40 mm)

`"c9"` (40 x 57 mm)

`"c8"` (57 x 81 mm)

`"c7"` (81 x 114 mm)

`"c6"` (114 x 162 mm)

`"c5"` (162 x 229 mm)

`"c4"` (229 x 324 mm)

`"c3"` (324 x 458 mm)

`"c2"` (458 x 648 mm)

`"c1"` (648 x 917 mm)

`"c0"` (917 x 1297 mm)

North American paper sizes

`"junior-legal"` (8.0 x 5.0 in)

`"legal"` (8.5 x 14.0 in)

`"ledger"` (17.0 x 11.0 in)

`"letter"` (8.5 x 11.0 in)

`"tabloid"`
(11.0 x 17.0 in)

`"11x17"` (11.0 x 17.0 in)

`"17x11"` (17.0 x 11.0 in)

Government-letter by IEEE Printer Working Group, for children's writing

`"government-letter"`
(8 x 10.5 in)

`"government-legal"`
(8.5 x 13.0 in)

`"philippine-legal"`
(8.5 x 13.0 in)

ANSI sizes

`"ansi a"` (8.5 x 11.0 in)

`"ansi b"` (17.0 x 11.0 in)

`"ansi c"` (17.0 x 22.0 in)

`"ansi d"` (22.0 x 34.0 in)

`"ansi e"` (34.0 x 44.0 in)

`"engineering f"`
(28.0 x 40.0 in)

North American Architectural sizes

`"arch a"` (9.0 x 12.0 in)

`"arch b"` (12.0 x 18.0 in)

`"arch c"` (18.0 x 24.0 in)

`"arch d"` (24.0 x 36.0 in)

`"arch e"` (36.0 x 48.0 in)

`"arch e1"` (30.0 x 42.0 in)

Antique sizes still used in the United Kingdom

`"statement"`
(5.5 x 8.5 in)

`"half letter"`
(5.5 x 8.5 in)

`"quarto"` (8.0 x 10.0 in)

`"octavo"` (6.75 x 10.5 in)

`"executive"`
(7.25 x 10.5 in)

`"monarch"`
(7.25 x 10.5 in)

`"foolscap"`
(8.27 x 13.0 in)

`"folio"` (8.27 x 13.0 in)

`"super-b"`
(13.0 x 19.0 in)

`"post"` (15.5 x 19.5 in)

`"crown"` (15.0 x 20.0 in)

`"large post"`
(16.5 x 21.0 in)

`"demy"` (17.5 x 22.5 in)

`"medium"` (18.0 x 23.0 in)

`"broadsheet"`
(18.0 x 24.0 in)

`"royal"` (20.0 x 25.0 in)

`"elephant"`
(23.0 x 28.0 in)

`"double demy"`
(22.5 x 35.0 in)

`"quad demy"`
(35.0 x 45.0 in)

`"atlas"` (26.0 x 34.0 in)

`"imperial"`
(22.0 x 30.0 in)

`"antiquarian"`
(31.0 x 53.0 in)

PA4 based sizes

`"pa0"` (840 x 1120 mm)

`"pa1"` (560 x 840 mm)

`"pa2"` (420 x 560 mm)

`"pa3"` (280 x 420 mm)

`"pa4"` (210 x 280 mm)

`"pa5"` (140 x 210 mm)

`"pa6"` (105 x 140 mm)

`"pa7"` (70 x 105 mm)

`"pa8"` (52 x 70 mm)

`"pa9"` (35 x 52 mm)

`"pa10"` (26 x 35 mm)

Used in Southeast Asia and Australia

`"f4"` (210 x 330 mm)

Used for very small `@lilypond` examples in the documentation based on a8 landscape.

`"a8landscape"`
(74 x 52 mm)

A.6 MIDI instruments

The following is a list of names that can be used for the `midiInstrument` property. The order of the instruments below, starting in the left-hand column moving down, corresponds to the General MIDI Standard's 128 Program Numbers.

```
acoustic grand
bright acoustic
electric grand
honky-tonk
electric piano 1
electric piano 2
harpsichord
clav
celesta
glockenspiel
music box
vibraphone
marimba
xylophone
tubular bells
dulcimer
drawbar organ
percussive organ
rock organ
church organ
reed organ
accordion
harmonica
concertina
acoustic guitar (nylon)
acoustic guitar (steel)
electric guitar (jazz)
electric guitar (clean)
electric guitar (muted)
overdriven guitar
distorted guitar
guitar harmonics
acoustic bass
electric bass (finger)
electric bass (pick)
fretless bass
slap bass 1
slap bass 2
synth bass 1
synth bass 2
violin
viola
cello
contrabass
tremolo strings
pizzicato strings
orchestral harp
timpani
string ensemble 1
string ensemble 2
synthstrings 1
synthstrings 2
choir aahs
voice oohs
synth voice
orchestra hit
trumpet
trombone
tuba
muted trumpet
french horn
brass section
synthbrass 1
synthbrass 2
soprano sax
alto sax
tenor sax
baritone sax
oboe
english horn
bassoon
clarinet
piccolo
flute
recorder
pan flute
blown bottle
shakuhachi
whistle
ocarina
lead 1 (square)
lead 2 (sawtooth)
lead 3 (calliope)
lead 4 (chiff)
lead 5 (charang)
lead 6 (voice)
lead 7 (fifths)
lead 8 (bass+lead)
pad 1 (new age)
pad 2 (warm)
pad 3 (polysynth)
pad 4 (choir)
pad 5 (bowed)
pad 6 (metallic)
pad 7 (halo)
pad 8 (sweep)
fx 1 (rain)
fx 2 (soundtrack)
fx 3 (crystal)
fx 4 (atmosphere)
fx 5 (brightness)
fx 6 (goblins)
fx 7 (echoes)
fx 8 (sci-fi)
sitar
banjo
shamisen
koto
kalimba
bagpipe
fiddle
shanai
tinkle bell
agogo
steel drums
woodblock
taiko drum
melodic tom
synth drum
reverse cymbal
guitar fret noise
breath noise
seashore
bird tweet
telephone ring
helicopter
applause
gunshot
```

A.7 List of colors

Normal colors

Usage syntax is detailed in ⟨undefined⟩ [Coloring objects], pagina ⟨undefined⟩.

```
black       white          red         green
blue        cyan           magenta     yellow
grey        darkred        darkgreen   darkblue
darkcyan    darkmagenta    darkyellow
```

X color names

X color names come several variants:

Any name that is spelled as a single word with capitalization (e.g. 'LightSlateBlue') can also be spelled as space separated words without capitalization (e.g. 'light slate blue').

The word 'grey' can always be spelled 'gray' (e.g. 'DarkSlateGray').

Some names can take a numerical suffix (e.g. 'LightSalmon4').

Color Names without a numerical suffix:

```
snow              GhostWhite       WhiteSmoke       gainsboro        FloralWhite
OldLace           linen            AntiqueWhite     PapayaWhip       BlanchedAlmond
bisque            PeachPuff        NavajoWhite      moccasin         cornsilk
ivory             LemonChiffon     seashell         honeydew         MintCream
azure             AliceBlue        lavender         LavenderBlush    MistyRose
white             black            DarkSlateGrey    DimGrey          SlateGrey
LightSlateGrey    grey             LightGrey        MidnightBlue     navy
NavyBlue          CornflowerBlue   DarkSlateBlue    SlateBlue        MediumSlateBlue
LightSlateBlue    MediumBlue       RoyalBlue        blue             DodgerBlue
DeepSkyBlue       SkyBlue          LightSkyBlue     SteelBlue        LightSteelBlue
LightBlue         PowderBlue       PaleTurquoise    DarkTurquoise    MediumTurquoise
turquoise         cyan             LightCyan        CadetBlue        MediumAquamarine
aquamarine        DarkGreen        DarkOliveGreen   DarkSeaGreen     SeaGreen
MediumSeaGreen    LightSeaGreen    PaleGreen        SpringGreen      LawnGreen
green             chartreuse       MediumSpringGreen       GreenYellow      LimeGreen
YellowGreen       ForestGreen      OliveDrab        DarkKhaki        khaki
PaleGoldenrod     LightGoldenrodYellow     LightYellow      yellow   gold
LightGoldenrod    goldenrod        DarkGoldenrod    RosyBrown        IndianRed
SaddleBrown       sienna           peru             burlywood        beige
wheat             SandyBrown       tan              chocolate        firebrick
brown             DarkSalmon       salmon           LightSalmon      orange
DarkOrange        coral            LightCoral       tomato           OrangeRed
red               HotPink          DeepPink         pink             LightPink
PaleVioletRed     maroon           MediumVioletRed  VioletRed        magenta
violet            plum             orchid           MediumOrchid     DarkOrchid
DarkViolet        BlueViolet       purple           MediumPurple     thistle
DarkGrey          DarkBlue         DarkCyan         DarkMagenta      DarkRed
LightGreen
```

Color names with a numerical suffix

In the following names the suffix N can be a number in the range 1-4:

```
snowN             seashellN        AntiqueWhiteN    bisqueN          PeachPuffN
NavajoWhiteN      LemonChiffonN    cornsilkN        ivoryN           honeydewN
LavenderBlushN    MistyRoseN       azureN           SlateBlueN       RoyalBlueN
blueN             DodgerBlueN      SteelBlueN       DeepSkyBlueN     SkyBlueN
```

LightSkyBlueN	LightSteelBlueN	LightBlueN	LightCyanN	PaleTurquoiseN
CadetBlueN	turquoiseN	cyanN	aquamarineN	DarkSeaGreenN
SeaGreenN	PaleGreenN	SpringGreenN	greenN	chartreuseN
OliveDrabN	DarkOliveGreenN	khakiN	LightGoldenrodN	LightYellowN
yellowN	goldN	goldenrodN	DarkGoldenrodN	RosyBrownN
IndianRedN	siennaN	burlywoodN	wheatN	tanN
chocolateN	firebrickN	brownN	salmonN	LightSalmonN
orangeN	DarkOrangeN	coralN	tomatoN	OrangeRedN
redN	DeepPinkN	HotPinkN	pinkN	LightPinkN
PaleVioletRedN	maroonN	VioletRedN	magentaN	orchidN
plumN	MediumOrchidN	DarkOrchidN	purpleN	MediumPurpleN
thistleN				

Grey Scale

A grey scale can be obtained using:

greyN

Where N is in the range 0-100.

A.8 The Feta font

The following symbols are available in the Emmentaler font and may be accessed directly using text markup with the name of the glyph as shown in the tables below, such as `g^\markup {\musicglyph #"scripts.segno" }` or `\markup {\musicglyph #"five"}`. For more information, see ⟨undefined⟩ [Formatting text], pagina ⟨undefined⟩.

Clef glyphs

clefs.C		clefs.C_change	
clefs.F		clefs.F_change	
clefs.G		clefs.G_change	
clefs.percussion		clefs.percussion_change	
clefs.tab		clefs.tab_change	

Time Signature glyphs

timesig.C44		timesig.C22	

Number glyphs

plus	+	comma	,
hyphen	-	period	.
zero	0	one	1
two	2	three	3
four	4	five	5
six	6	seven	7
eight	8	nine	9

Accidental glyphs

accidentals.sharp	♯	accidentals .sharp.arrowup	♯
accidentals .sharp.arrowdown	♯	accidentals .sharp.arrowboth	♯
accidentals.sharp .slashslash.stem	♯	accidentals.sharp .slashslashslash.stemstem	♯
accidentals.sharp .slashslashslash.stem	♯	accidentals.sharp .slashslash.stemstemstem	♯
accidentals.natural	♮	accidentals .natural.arrowup	♮
accidentals .natural.arrowdown	♮	accidentals .natural.arrowboth	♮
accidentals.flat	♭	accidentals.flat.arrowup	♭

accidentals .flat.arrowdown		accidentals .flat.arrowboth	
accidentals.flat.slash		accidentals.flat .slashslash	
accidentals .mirroredflat.flat		accidentals.mirroredflat	
accidentals .mirroredflat.backslash		accidentals.flatflat	
accidentals .flatflat.slash		accidentals.doublesharp	
accidentals.rightparen		accidentals.leftparen	

Default Notehead glyphs

noteheads.uM2		noteheads.dM2	
noteheads.sM1		noteheads.s0	
noteheads.s1		noteheads.s2	

Special Notehead glyphs

noteheads.sM1double		noteheads.s0diamond	
noteheads.s1diamond		noteheads.s2diamond	
noteheads.s0triangle		noteheads.d1triangle	
noteheads.u1triangle		noteheads.u2triangle	
noteheads.d2triangle		noteheads.s0slash	

noteheads.s1slash		noteheads.s2slash	
noteheads.s0cross		noteheads.s1cross	
noteheads.s2cross		noteheads.s2xcircle	
noteheads.s0harmonic		noteheads.s2harmonic	

Shape-note Notehead glyphs

noteheads.s0do		noteheads.d1do	
noteheads.u1do		noteheads.d2do	
noteheads.u2do		noteheads.s0doThin	
noteheads.d1doThin		noteheads.u1doThin	
noteheads.d2doThin		noteheads.u2doThin	
noteheads.s0re		noteheads.u1re	
noteheads.d1re		noteheads.u2re	
noteheads.d2re		noteheads.s0reThin	
noteheads.u1reThin		noteheads.d1reThin	
noteheads.u2reThin		noteheads.d2reThin	
noteheads.s0mi		noteheads.s1mi	

noteheads.s2mi

noteheads.s0miMirror

noteheads.s1miMirror

noteheads.s2miMirror

noteheads.s0miThin

noteheads.s1miThin

noteheads.s2miThin

noteheads.u0fa

noteheads.d0fa

noteheads.u1fa

noteheads.d1fa

noteheads.u2fa

noteheads.d2fa

noteheads.u0faThin

noteheads.d0faThin

noteheads.u1faThin

noteheads.d1faThin

noteheads.u2faThin

noteheads.d2faThin

noteheads.s0sol 

noteheads.s1sol 

noteheads.s2sol 

noteheads.s0la

noteheads.s1la

noteheads.s2la

noteheads.s0laThin

noteheads.s1laThin

noteheads.s2laThin

noteheads.s0ti

noteheads.u1ti

noteheads.d1ti		noteheads.u2ti	
noteheads.d2ti		noteheads.s0tiThin	
noteheads.u1tiThin		noteheads.d1tiThin	
noteheads.u2tiThin		noteheads.d2tiThin	
noteheads.u0doFunk		noteheads.d0doFunk	
noteheads.u1doFunk		noteheads.d1doFunk	
noteheads.u2doFunk		noteheads.d2doFunk	
noteheads.u0reFunk		noteheads.d0reFunk	
noteheads.u1reFunk		noteheads.d1reFunk	
noteheads.u2reFunk		noteheads.d2reFunk	
noteheads.u0miFunk		noteheads.d0miFunk	
noteheads.u1miFunk		noteheads.d1miFunk	
noteheads.s2miFunk		noteheads.u0faFunk	
noteheads.d0faFunk		noteheads.u1faFunk	
noteheads.d1faFunk		noteheads.u2faFunk	

noteheads.d2faFunk		noteheads.s0solFunk	
noteheads.s1solFunk		noteheads.s2solFunk	
noteheads.s0laFunk		noteheads.s1laFunk	
noteheads.s2laFunk		noteheads.u0tiFunk	
noteheads.d0tiFunk		noteheads.u1tiFunk	
noteheads.d1tiFunk		noteheads.u2tiFunk	
noteheads.d2tiFunk		noteheads.s0doWalker	
noteheads.u1doWalker		noteheads.d1doWalker	
noteheads.u2doWalker		noteheads.d2doWalker	
noteheads.s0reWalker		noteheads.u1reWalker	
noteheads.d1reWalker		noteheads.u2reWalker	
noteheads.d2reWalker		noteheads.s0miWalker	
noteheads.s1miWalker		noteheads.s2miWalker	
noteheads.s0faWalker		noteheads.u1faWalker	
noteheads.d1faWalker		noteheads.u2faWalker	

noteheads.d2faWalker

noteheads.s0laWalker

noteheads.s1laWalker

noteheads.s2laWalker

noteheads.s0tiWalker

noteheads.u1tiWalker

noteheads.d1tiWalker

noteheads.u2tiWalker

noteheads.d2tiWalker

Rest glyphs

rests.0

rests.1

rests.0o

rests.1o

rests.M3

rests.M2

rests.M1

rests.M1o

rests.2

rests.2classical

rests.3

rests.4

rests.5

rests.6

rests.7

Flag glyphs

flags.u3		flags.u4	
flags.u5		flags.u6	
flags.u7		flags.d3	
flags.d4		flags.d5	
flags.d6		flags.d7	
flags.ugrace		flags.dgrace	

Dot glyphs

dots.dot	

Dynamic glyphs

space		f	
m		p	
r		s	
z			

Script glyphs

scripts.ufermata		scripts.dfermata	
scripts.ushortfermata		scripts.dshortfermata	
scripts.ulongfermata		scripts.dlongfermata	
scripts.uverylongfermata		scripts.dverylongfermata	
scripts.thumb		scripts.sforzato	
scripts.espr		scripts.staccato	
scripts.ustaccatissimo		scripts.dstaccatissimo	
scripts.tenuto		scripts.uportato	
scripts.dportato		scripts.umarcato	
scripts.dmarcato		scripts.open	
scripts.halfopen		scripts.halfopenvertical	
scripts.stopped		scripts.upbow	
scripts.downbow		scripts.reverseturn	
scripts.turn		scripts.trill	

scripts.upedalheel 

scripts.dpedalheel 

scripts.upedaltoe 

scripts.dpedaltoe 

scripts.flageolet

scripts.segno 

scripts.varsegno 

scripts.coda 

scripts.varcoda 

scripts.rcomma

scripts.lcomma

scripts.rvarcomma 

scripts.lvarcomma 

scripts.arpeggio 

scripts.trill_element

scripts.arpeggio
.arrow.M1 

scripts.arpeggio.arrow.1 

scripts.trilelement

scripts.prall

scripts.mordent 

scripts.prallprall

scripts.prallmordent 

scripts.upprall 

scripts.upmordent 

scripts.pralldown 

scripts.downprall 

scripts.downmordent 

scripts.prallup 

scripts.lineprall 

scripts.caesura.curved 

scripts.caesura.straight		scripts.tickmark	
scripts.snappizzicato		scripts.ictus	
scripts.uaccentus		scripts.daccentus	
scripts.usemicirculus		scripts.dsemicirculus	
scripts.circulus		scripts.augmentum	
scripts .usignumcongruentiae		scripts .dsignumcongruentiae	

Arrowhead glyphs

arrowheads.open.01		arrowheads.open.0M1	
arrowheads.open.11		arrowheads.open.1M1	
arrowheads.close.01		arrowheads.close.0M1	
arrowheads.close.11		arrowheads.close.1M1	

Bracket-tip glyphs

brackettips.up		brackettips.down	

Pedal glyphs

pedal.*		pedal.M	
pedal..		pedal.P	
pedal.d		pedal.e	
pedal.Ped			

Accordion glyphs

accordion.discant		accordion.dot	
accordion.freebass		accordion.stdbass	
accordion.bayanbass		accordion.oldEE	
accordion.push		accordion.pull	

Tie glyphs

ties.lyric.short		ties.lyric.default	

Vaticana glyphs

clefs.vaticana.do		clefs.vaticana.do_change	
clefs.vaticana.fa		clefs.vaticana.fa_change	
custodes.vaticana.u0		custodes.vaticana.u1	
custodes.vaticana.u2		custodes.vaticana.d0	
custodes.vaticana.d1		custodes.vaticana.d2	
accidentals.vaticanaM1		accidentals.vaticana0	
dots.dotvaticana		noteheads .svaticana.punctum	
noteheads.svaticana .punctum.cavum		noteheads.svaticana .linea.punctum	

noteheads.svaticana .linea.punctum.cavum	noteheads.svaticana .inclinatum
noteheads.svaticana.lpes	noteheads .svaticana.vlpes
noteheads.svaticana.upes	noteheads .svaticana.vupes
noteheads .svaticana.plica	noteheads .svaticana.vplica
noteheads .svaticana.epiphonus	noteheads.svaticana .vepiphonus
noteheads.svaticana .reverse.plica	noteheads.svaticana .reverse.vplica
noteheads.svaticana .inner.cephalicus	noteheads.svaticana .cephalicus
noteheads .svaticana.quilisma	

Medicaea glyphs

clefs.medicaea.do	clefs.medicaea.do_change
clefs.medicaea.fa	clefs.medicaea.fa_change
custodes.medicaea.u0	custodes.medicaea.u1
custodes.medicaea.u2	custodes.medicaea.d0
custodes.medicaea.d1	custodes.medicaea.d2
accidentals.medicaeaM1	noteheads.smedicaea .inclinatum
noteheads .smedicaea.punctum	noteheads .smedicaea.rvirga
noteheads .smedicaea.virga	

Hufnagel glyphs

clefs.hufnagel.do 

clefs.hufnagel.do_change 

clefs.hufnagel.fa 

clefs.hufnagel.fa_change 

clefs.hufnagel.do.fa 

clefs.hufnagel
.do.fa_change 

custodes.hufnagel.u0 

custodes.hufnagel.u1 

custodes.hufnagel.u2 

custodes.hufnagel.d0 

custodes.hufnagel.d1 

custodes.hufnagel.d2 

accidentals.hufnagelM1 

noteheads
.shufnagel.punctum

noteheads
.shufnagel.virga 

noteheads.shufnagel.lpes

Mensural glyphs

rests.M3mensural

rests.M2mensural

rests.M1mensural

rests.0mensural

rests.1mensural

rests.2mensural

rests.3mensural

rests.4mensural

clefs.mensural.c 

clefs.mensural.c_change 

clefs.blackmensural.c 

clefs.blackmensural
.c_change 

clefs.mensural.f 

clefs.mensural.f_change 

clefs.mensural.g 

clefs.mensural.g_change 

custodes.mensural.u0 

custodes.mensural.u1 

custodes.mensural.u2 

custodes.mensural.d0 

custodes.mensural.d1 

custodes.mensural.d2 

accidentals.mensural1

accidentals.mensuralM1

flags.mensuralu03 

flags.mensuralu13

flags.mensuralu23 

flags.mensurald03

flags.mensurald13

flags.mensurald23

flags.mensuralu04 

flags.mensuralu14

flags.mensuralu24 

flags.mensurald04

flags.mensurald14

flags.mensurald24 

flags.mensuralu05 

flags.mensuralu15

flags.mensuralu25 

flags.mensurald05 

flags.mensurald15

flags.mensurald25 

flags.mensuralu06 

flags.mensuralu16

flags.mensuralu26 

flags.mensurald06 

flags.mensurald16

flags.mensurald26 

timesig.mensural44 

timesig.mensural22 

timesig.mensural32 

timesig.mensural64 

timesig.mensural94 

timesig.mensural34 

timesig.mensural68 

timesig.mensural98 

timesig.mensural48 

timesig.mensural68alt 

timesig.mensural24 

noteheads.uM3mensural 

noteheads.dM3mensural 

noteheads.sM3ligmensural 

noteheads.uM2mensural 

noteheads.dM2mensural 

noteheads.sM2ligmensural 

noteheads.sM1mensural 

noteheads.urM3mensural 

noteheads.drM3mensural 

noteheads
.srM3ligmensural 

noteheads.urM2mensural 

noteheads.drM2mensural 

noteheads
.srM2ligmensural 

noteheads.srM1mensural 

noteheads
.uM3semimensural 

noteheads
.dM3semimensural 

noteheads
.sM3semiligmensural 

noteheads
.uM2semimensural 

noteheads
.dM2semimensural 

noteheads
.sM2semiligmensural 

noteheads
.sM1semimensural 

noteheads
.urM3semimensural 

noteheads
.drM3semimensural 

noteheads
.srM3semiligmensural 

noteheads
.urM2semimensural 

noteheads
.drM2semimensural 

noteheads
.srM2semiligmensural 

noteheads
.srM1semimensural 

noteheads
.uM3blackmensural 

noteheads
.dM3blackmensural 

noteheads
.sM3blackligmensural 

noteheads
.uM2blackmensural 

noteheads
.dM2blackmensural 

noteheads
.sM2blackligmensural 

noteheads
.sM1blackmensural 

noteheads.s0mensural

noteheads.s1mensural

noteheads.s2mensural

noteheads
.s0blackmensural

Neomensural glyphs

rests.M3neomensural		rests.M2neomensural	
rests.M1neomensural		rests.0neomensural	
rests.1neomensural		rests.2neomensural	
rests.3neomensural		rests.4neomensural	
clefs.neomensural.c		clefs.neomensural .c_change	
timesig.neomensural44		timesig.neomensural22	
timesig.neomensural32		timesig.neomensural64	
timesig.neomensural94		timesig.neomensural34	
timesig.neomensural68		timesig.neomensural98	
timesig.neomensural48		timesig.neomensural68alt	
timesig.neomensural24		noteheads.uM3neomensural	
noteheads.dM3neomensural		noteheads.uM2neomensural	
noteheads.dM2neomensural		noteheads.sM1neomensural	
noteheads .urM3neomensural		noteheads .drM3neomensural	

noteheads .urM2neomensural 

noteheads .drM2neomensural 

noteheads .srM1neomensural 

noteheads.s0neomensural 

noteheads.s1neomensural 

noteheads.s2neomensural 

Petrucci glyphs

clefs.petrucci.c1 

clefs.petrucci.c1_change 

clefs.petrucci.c2 

clefs.petrucci.c2_change 

clefs.petrucci.c3 

clefs.petrucci.c3_change 

clefs.petrucci.c4 

clefs.petrucci.c4_change 

clefs.petrucci.c5 

clefs.petrucci.c5_change 

clefs.petrucci.f 

clefs.petrucci.f_change 

clefs.petrucci.g 

clefs.petrucci.g_change 

noteheads.s0petrucci 

noteheads.s1petrucci 

noteheads.s2petrucci 

noteheads .s0blackpetrucci 

noteheads .s1blackpetrucci 

noteheads .s2blackpetrucci 

Solesmes glyphs

noteheads.ssolesmes.incl.parvum

noteheads.ssolesmes.auct.asc

noteheads.ssolesmes.auct.desc

noteheads.ssolesmes.incl.auctum

noteheads.ssolesmes.stropha

noteheads.ssolesmes.stropha.aucta

noteheads.ssolesmes.oriscus

Kievan Notation glyphs

clefs.kievan.do

clefs.kievan.do_change

accidentals.kievan1

accidentals.kievanM1

scripts.barline.kievan

dots.dotkievan

noteheads.sM2kievan

noteheads.sM1kievan

noteheads.s0kievan

noteheads.d2kievan

noteheads.u2kievan

noteheads.s1kievan

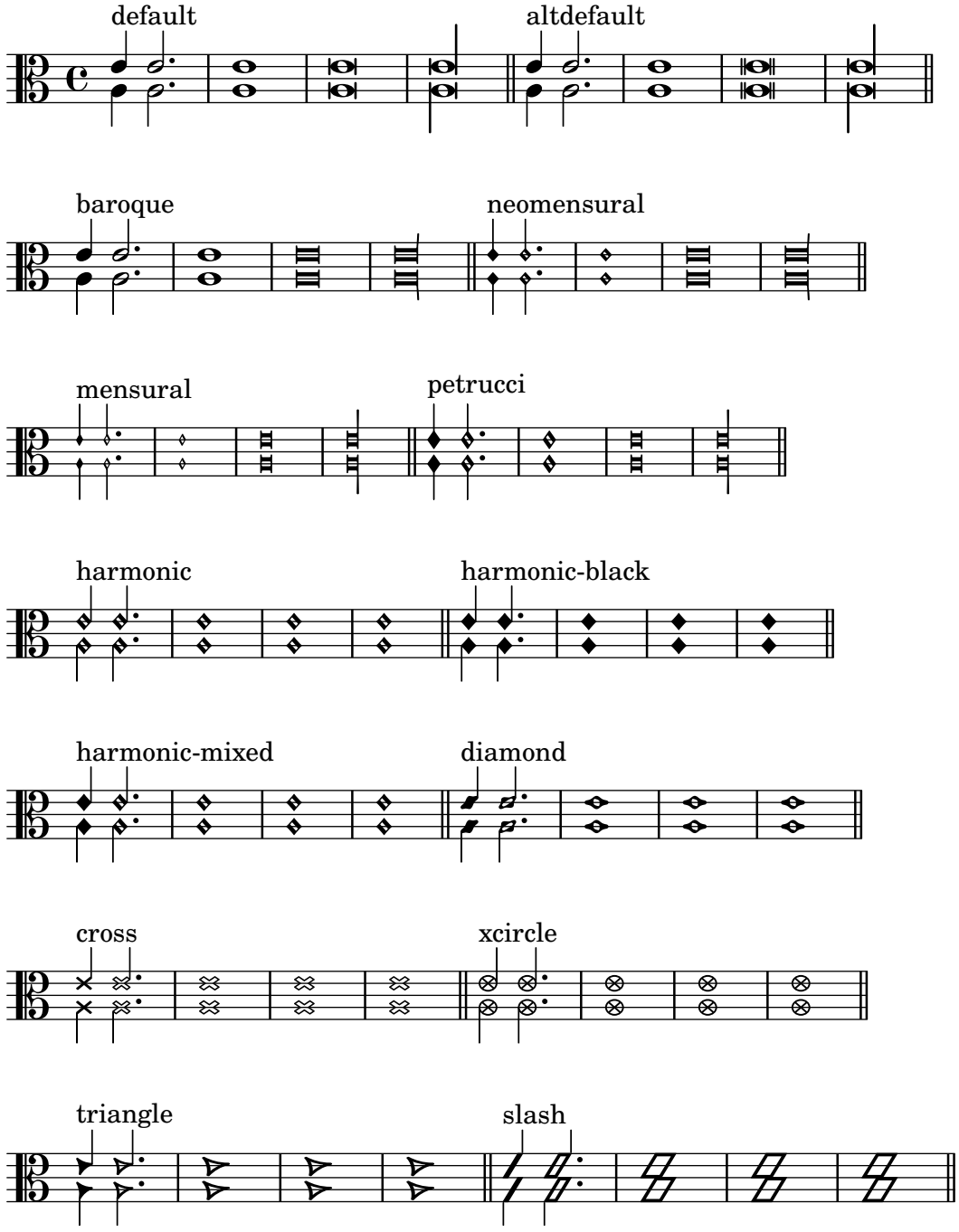
noteheads.sr1kievan

noteheads.d3kievan

noteheads.u3kievan

A.9 Note head styles

The following styles may be used for note heads.



A.10 Text markup commands

The following commands can all be used inside `\markup { }`.

A.10.1 Font

`\abs-fontsize` *size* (number) *arg* (markup)

Use *size* as the absolute font size to display *arg*. Adjusts `baseline-skip` and `word-space` accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```

default text font size text font size 16 text font size 12

`\bold` *arg* (markup)
: Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```

default **bold**

`\box` *arg* (markup)
: Draw a box round *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \override #'(box-padding . 0.5)
  \box
  \line { V. S. }
}
```

V. S.

Used properties:

- `box-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\caps` *arg* (markup)
: Copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\dynamic` *arg* (markup)
: Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like 'più **f**', the normal words (like 'più') should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

sfzp

`\finger` *arg* (markup)
Set *arg* as small numbers.

```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```

1 2 3 4 5

`\fontCaps` *arg* (markup)
Set `font-shape` to `caps`

Note: `\fontCaps` requires the installation and selection of fonts which support the `caps` font shape.

`\fontsize` *increment* (number) *arg* (markup)
Add *increment* to the font-size. Adjusts `baseline-skip` accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5
  smaller
}
```

default smaller

Used properties:

- `baseline-skip` (2)
- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)
Set font size to +2.

```
\markup {
  default
  \hspace #2
  \huge
  huge
}
```

default huge

`\italic` *arg* (markup)
Use italic `font-shape` for *arg*.

```
\markup {
  default
  \hspace #2
  \italic
  italic
}
```

default *italic*

`\large` *arg* (markup)
Set font size to +1.

```
\markup {
  default
  \hspace #2
  \large
  large
}
```

default large

`\larger` *arg* (markup)
Increase the font size relative to the current setting.

```
\markup {
  default
  \hspace #2
  \larger
  larger
}
```

default larger

`\magnify` *sz* (number) *arg* (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:

```
A \magnify #1.1 { A } A
```

Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

default 50% larger

`\medium` *arg* (markup)
Switch to medium font-series (in contrast to bold).

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}
```

some bold text medium font series **bold again**

`\normal-size-sub` *arg* (markup)
Set *arg* in subscript with a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default $_{\text{subscript in standard size}}$

Used properties:

- `baseline-skip`

`\normal-size-super` *arg* (markup)
Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

default $^{\text{superscript in standard size}}$

Used properties:

- `baseline-skip`

`\normal-text` *arg* (markup)
Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \caps {
    huge bold sans caps
    \hspace #2
    \normal-text {
      huge normal
    }
```

```
    \hspace #2
    as before
  }
}
```

HUGE BOLD SANS CAPS huge normal **AS BEFORE**

`\normalsize` *arg* (markup)
Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

this is very small normal size teeny again

`\number` *arg* (markup)
Set font family to `number`, which yields the font used for time signatures and fingerings. This font contains numbers and some punctuation; it has no letters.

```
\markup {
  \number {
    0 1 2 3 4 5 6 7 8 9 . ,
  }
}
```

0123456789.,

`\replace` *replacements* (list) *arg* (markup)
Used to automatically replace a string by another in the markup *arg*. Each pair of the alist *replacements* specifies what should be replaced. The `key` is the string to be replaced by the `value` string.

```
\markup \replace #'(("thx" . "Thanks!")) thx
```

Thanks!

`\roman` *arg* (markup)
Set font family to `roman`.

```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
```

```
    \hspace #2
    return to sans
  }
}
```

sans serif, bold text in roman font family return to sans

`\sans` *arg* (markup)
Switch to the sans serif font family.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```

default sans serif

`\simple` *str* (string)
A simple text string; `\markup { foo }` is equivalent with `\markup { \simple #"foo" }`.

Note: for creating standard text markup or defining new markup commands, the use of `\simple` is unnecessary.

```
\markup {
  \simple #"simple"
  \simple #"text"
  \simple #"strings"
}
```

simple text strings

`\small` *arg* (markup)
Set font size to -1.

```
\markup {
  default
  \hspace #2
  \small
  small
}
```

default small

`\smallCaps` *arg* (markup)
Emit *arg* as small caps.

Note: `\smallCaps` does not support accented characters.

```
\markup {
  default
  \hspace #2
  \smallCaps {
    Text in small caps
```

```
  }
}
```

default TEXT IN SMALL CAPS

`\smaller` *arg* (markup)
Decrease the font size relative to the current setting.

```
\markup {
  \fontsize #3.5 {
    some large text
    \hspace #2
    \smaller {
      a bit smaller
    }
    \hspace #2
    more large text
  }
}
```

some large text a bit smaller more large text

`\sub` *arg* (markup)
Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    O
  }
}
```

H_2O

Used properties:

- `baseline-skip`
- `font-size` (0)

`\super` *arg* (markup)
Set *arg* in superscript.

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

$E = mc^2$

Used properties:

- `baseline-skip`
- `font-size` (0)

`\teeny` *arg* (markup)
Set font size to -3.

```
\markup {
  default
  \hspace #2
  \teeny
  teeny
}
```

default teeny

`\text` *arg* (markup)
Use a text font instead of music symbol or music alphabet font.

```
\markup {
  \number {
    1, 2,
    \text {
      three, four,
    }
    5
  }
}
```

1, 2, three, four, **5**

`\tiny` *arg* (markup)
Set font size to -2.

```
\markup {
  default
  \hspace #2
  \tiny
  tiny
}
```

default tiny

`\typewriter` *arg* (markup)
Use `font-family` typewriter for *arg*.

```
\markup {
  default
  \hspace #2
  \typewriter
  typewriter
}
```

default typewriter

`\underline` *arg* (markup)
Underline *arg*. Looks at `thickness` to determine line thickness, and `offset` to determine line y-offset.

```
\markup \fill-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
}
```

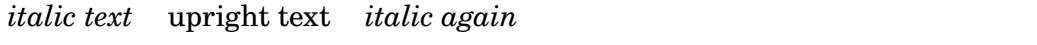


Used properties:

- `offset` (2)
- `thickness` (1)

`\upright` *arg* (markup)
: Set `font-shape` to `upright`. This is the opposite of `italic`.

```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
    }
    \hspace #2
    italic again
  }
}
```



A.10.2 Align

`\center-align` *arg* (markup)
: Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```



`\center-column` *args* (markup list)
: Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\column` *args* (markup list)

Stack the markups in *args* vertically. The property `baseline-skip` determines the space between markups in *args*.

```
\markup {
  \column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\combine` *arg1* (markup) *arg2* (markup)

Print two markups on top of each other.

Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; the follow example will not compile:

```
\combine { a list }
```

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}
```



`\concat` *args* (markup list)

Concatenate *args* in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to `"fi"`.

```
\markup {
  \concat {
    one
    two
    three
  }
}
```

onetwothree

`\dir-column` *args* (markup list)
Make a column of *args*, going up or down, depending on the setting of the `direction` layout property.

```
\markup {
  \override #`(direction . ,UP) {
    \dir-column {
      going up
    }
  }
  \hspace #1
  \dir-column {
    going down
  }
  \hspace #1
  \override #'(direction . 1) {
    \dir-column {
      going up
    }
  }
}
```

up
going
up
going
down
going

Used properties:

- `baseline-skip`
- `direction`

`\fill-line` *args* (markup list)
Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

```
\markup {
  \column {
    \fill-line {
      Words evenly spaced across the page
    }
    \null
    \fill-line {
      \line { Text markups }
      \line {
        \italic { evenly spaced }
```

```
      }
      \line { across the page }
    }
  }
}
```

Words evenly spaced across the page

Text markups *evenly spaced* across the page

Used properties:

- `line-width` (#f)
- `word-space` (0.6)
- `text-direction` (1)

`\fill-with-pattern` *space* (number) *dir* (direction) *pattern* (markup) *left* (markup) *right* (markup)

Put *left* and *right* in a horizontal line of width `line-width` with a line of markups *pattern* in between. Patterns are spaced apart by *space*. Patterns are aligned to the *dir* markup.

```
\markup \column {
  "right-aligned :"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  \null
  "center-aligned :"
  \fill-with-pattern #1.5 #CENTER - left right
  \null
  "left-aligned :"
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left first
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left second
}
```

right-aligned :
first . . . . . right
second . . . . . right

center-aligned :
left - right

left-aligned :
left : : : : : : : : : : : : : : : : : first
left : : : : : : : : : : : : : : : : : second

Used properties:

- `line-width`
- `word-space`

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)

Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT
    two
    three
    \null
    one
    \general-align #X #CENTER
    two
    three
    \null
    \line {
      one
      \general-align #Y #UP
      two
      three
    }
    \null
    \line {
      one
      \general-align #Y #3.2
      two
      three
    }
  }
}
```

one
two
three

one
two
three

one two three

one three
two

`\halign` *dir* (number) *arg* (markup)
Set horizontal alignment. If *dir* is `-1`, then it is left-aligned, while `+1` is right. Values in between interpolate alignment accordingly.

```
\markup {
  \column {
    one
    \halign #LEFT
    two
    three
    \null
```

```
    one
    \halign #CENTER
    two
    three
    \null
    one
    \halign #RIGHT
    two
    three
    \null
    one
    \halign #-5
    two
    three
  }
}
```

one
two
three

one
two
three
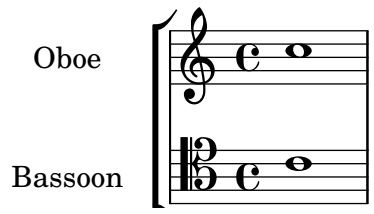
one
two
three

one
two
three

`\hcenter-in` *length* (number) *arg* (markup)
: Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Bassoon
    }
    \clef tenor
    c'1
```

```
  }
>>
```



`\hspace` *amount* (number)

Create an invisible object taking up horizontal space *amount*.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one two three

`\justify-field` *symbol* (symbol)

Justify the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et dolore magna
    aliqua.  Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\justify` *args* (markup list)

Like `\wordwrap`, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; X is the number of staff spaces.

```
\markup {
  \justify {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\justify-string` *arg* (string)

Justify a string. Paragraphs may be separated with double newlines

```
\markup {
  \override #'(line-width . 40)
  \justify-string #"Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut labore
      et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat.


      Excepteur sint occaecat cupidatat non proident, sunt in culpa
      qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

`\left-align` *arg* (markup)
Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align
    two
    three
  }
}
```

one
two
three

`\left-column` *args* (markup list)
Put `args` in a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

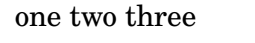
one
two
three

Used properties:

- `baseline-skip`

`\line` *args* (markup list)
Put *args* in a horizontal line. The property `word-space` determines the space between markups in *args*.

```
\markup {
  \line {
    one two three
  }
}
```

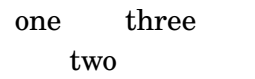


Used properties:

- `text-direction` (1)
- `word-space`

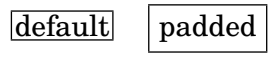
`\lower` *amount* (number) *arg* (markup)
: Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.

```
\markup {
  one
  \lower #3
  two
  three
}
```



`\pad-around` *amount* (number) *arg* (markup)
: Add padding *amount* all around *arg*.

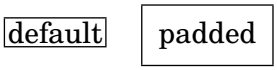
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```



`\pad-markup` *amount* (number) *arg* (markup)
: Add space around a markup object. Identical to `pad-around`.

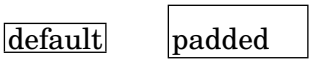
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-markup #1 {
      padded
```

```
    }
  }
}
```



`\pad-to-box` *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)
Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```



`\pad-x` *amount* (number) *arg* (markup)
Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```



`\put-adjacent` *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)
Put *arg2* next to *arg1*, without moving *arg1*.

`\raise` *amount* (number) *arg* (markup)
Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

```
\markup {
  C
  \small
  \bold
  \raise #1.0
  9/7+
}
```

C **9/7+**

`\right-align` *arg* (markup)
Align *arg* on its right edge.

```
\markup {
  \column {
    one
    \right-align
    two
    three
  }
}
```

one
two
three

`\right-column` *args* (markup list)
Put `args` in a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\rotate` *ang* (number) *arg* (markup)
Rotate object with *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
  \line {
    rotated 45˚
  }
}
```

default rotated 45°

`\translate` *offset* (pair of numbers) *arg* (markup)
Translate *arg* relative to its surroundings. *offset* is a pair of numbers representing the displacement in the X and Y axis.

```
\markup {
  *
  \translate #'(2 . 3)
  \line { translated two spaces right, three up }
}
```

translated two spaces right, three up
*

`\translate-scaled` *offset* (pair of numbers) *arg* (markup)
Translate *arg* by *offset*, scaling the offset by the `font-size`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```

* translate * translate-scaled

Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)
Align `arg` to its Y center.

```
\markup {
  one
  \vcenter
  two
  three
}
```

one two three

`\vspace` *amount* (number)
Create an invisible object taking up vertical space of *amount* multiplied by 3.

```
\markup {
    \center-column {
    one
    \vspace #2
    two
    \vspace #5
    three
  }
}
```

one

two

three

`\wordwrap-field` *symbol* (symbol)
Wordwrap the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea commodo
    consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\wordwrap` *args* (markup list)
Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where X is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
```

```
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string` *arg* (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string #"Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut labore
      et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat.


      Excepteur sint occaecat cupidatat non proident, sunt in culpa
      qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et
dolore magna aliqua.
Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo
consequat.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia
deserunt mollit anim id est laborum

Used properties:

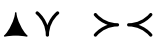
- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

A.10.3 Graphic

`\arrow-head` *axis* (integer) *dir* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

```
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}
```



`\beam` *width* (number) *slope* (number) *thickness* (number)

Create a beam with the specified parameters.

```
\markup {
  \beam #5 #1 #2
}
```



`\bracket` *arg* (markup)

Draw vertical brackets around *arg*.

```
\markup {
  \bracket {
    \note #"2." #UP
  }
}
```



`\circle` *arg* (markup)

Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \circle {
    Hi
  }
}
```



Used properties:

- `circle-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\draw-circle` *radius* (number) *thickness* (number) *filled* (boolean)

A circle of radius *radius* and thickness *thickness*, optionally filled.

```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}
```

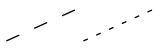


`\draw-dashed-line` *dest* (pair of numbers)

A dashed line.

If `full-length` is set to #t (default) the dashed-line extends to the whole length given by *dest*, without white space at beginning or end. `off` will then be altered to fit. To insist on the given (or default) values of `on`, `off` use `\override #'(full-length . #f)` Manual settings for `on`,`off` and `phase` are possible.

```
\markup {
  \draw-dashed-line #'(5.1 . 2.3)
  \override #'(on . 0.3)
  \override #'(off . 0.5)
  \draw-dashed-line #'(5.1 . 2.3)
}
```



Used properties:

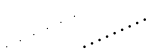
- `full-length` (#t)
- `phase` (0)
- `off` (1)
- `on` (1)
- `thickness` (1)

`\draw-dotted-line` *dest* (pair of numbers)

A dotted line.

The dotted-line always extends to the whole length given by *dest*, without white space at beginning or end. Manual settings for `off` are possible to get larger or smaller space between the dots. The given (or default) value of `off` will be altered to fit the line-length.

```
\markup {
  \draw-dotted-line #'(5.1 . 2.3)
  \override #'(thickness . 2)
  \override #'(off . 0.2)
  \draw-dotted-line #'(5.1 . 2.3)
}
```

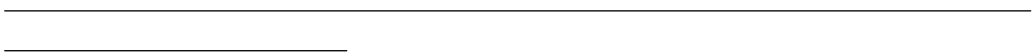


Used properties:

- `phase` (0)
- `off` (1)
- `thickness` (1)

`\draw-hline`
: Draws a line across a page, where the property `span-factor` controls what fraction of the page is taken up.

```
\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}
```

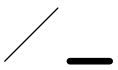


Used properties:

- `span-factor` (1)
- `line-width`
- `draw-line-markup`

`\draw-line` *dest* (pair of numbers)
: A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```



Used properties:

- `thickness` (1)

`\ellipse` *arg* (markup)
: Draw an ellipse around *arg*. Use `thickness`, `x-padding`, `y-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \ellipse {
    Hi
  }
}
```



Used properties:

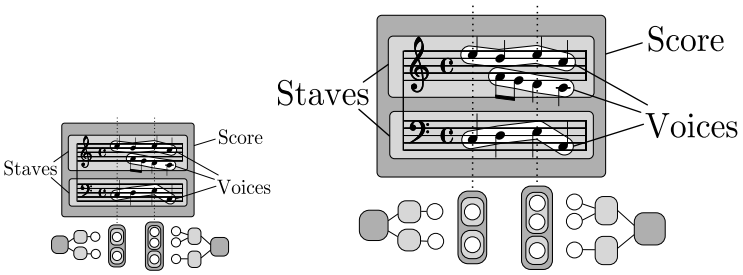
- `y-padding` (0.2)
- `x-padding` (0.2)
- `font-size` (0)

- `thickness` (1)

`\epsfile` *axis* (number) *size* (number) *file-name* (string)

Inline an EPS image. The image is scaled along *axis* to *size*.

```
\markup {
  \general-align #Y #DOWN {
    \epsfile #X #20 #"context-example.eps"
    \epsfile #Y #20 #"context-example.eps"
  }
}
```



`\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)

Draw a box with rounded corners of dimensions *xext* and *yext*. For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).

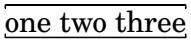
```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \filled-box #'(1 . 8) #'(0 . 7) #0.2
  \with-color #white
  \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
}
```



`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}
```



`\oval` *arg* (markup)

Draw an oval around *arg*. Use `thickness`, `x-padding`, `x-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \oval {
    Hi
  }
}
```



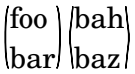
Used properties:

- `y-padding` (0.75)
- `x-padding` (0.75)
- `font-size` (0)
- `thickness` (1)

`\parenthesize` *arg* (markup)

Draw parentheses around *arg*. This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \line {
    \parenthesize {
      \column {
        foo
        bar
      }
    }
    \override #'(angularity . 2) {
      \parenthesize {
        \column {
          bah
          baz
        }
      }
    }
  }
}
```



Used properties:

- `width` (0.25)
- `thickness` (1)
- `size` (1)
- `padding`
- `angularity` (0)

`\path` *thickness* (number) *commands* (list)

Draws a path with line *thickness* according to the directions given in *commands*. *commands* is a list of lists where the `car` of each sublist is a drawing command and the `cdr` comprises the associated arguments for each command.

There are seven commands available to use in the list `commands`: `moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `rcurveto`, and `closepath`. Note that the commands that begin with *r* are the relative variants of the other three commands.

The commands `moveto`, `rmoveto`, `lineto`, and `rlineto` take 2 arguments; they are the X and Y coordinates for the destination point.

The commands `curveto` and `rcurveto` create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The `closepath` command takes zero arguments and closes the current subpath in the active path.

Note that a sequence of commands *must* begin with a `moveto` or `rmoveto` to work with the SVG output.

Line-cap styles and line-join styles may be customized by overriding the `line-cap-style` and `line-join-style` properties, respectively. Available line-cap styles are `'butt`, `'round`, and `'square`. Available line-join styles are `'miter`, `'round`, and `'bevel`.

The property `filled` specifies whether or not the path is filled with color.

```
samplePath =
  #'((moveto 0 0)
     (lineto -1 1)
     (lineto 1 1)
     (lineto 1 -1)
     (curveto -5 -5 -5 5 -1 0)
     (closepath))

\markup {
  \path #0.25 #samplePath

  \override #'(line-join-style . miter) \path #0.25 #samplePath

  \override #'(filled . #t) \path #0.25 #samplePath
}
```



Used properties:

- `filled` (#f)
- `line-join-style` (round)
- `line-cap-style` (round)

`\postscript` *str* (string)
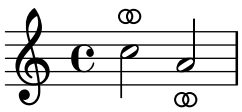
This inserts *str* directly into the output as a PostScript command string.

```
ringsps = #"
  0.15 setlinewidth
  0.9 0.6 moveto
  0.4 0.6 0.5 0 361 arc
  stroke
  1.0 0.6 0.5 0 361 arc
  stroke
```

```
  "

rings = \markup {
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}
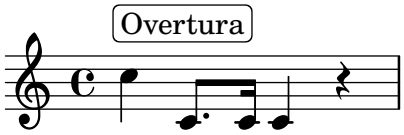
\relative c'' {
  c2^\rings
  a2_\rings
}
```



`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup; the `corner-radius` property makes it possible to define another shape for the corners (default is 1).

```
c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r
```



Used properties:

- `box-padding` (0.5)
- `font-size` (0)
- `corner-radius` (1)
- `thickness` (1)

`\scale` *factor-pair* (pair of numbers) *arg* (markup)

Scale *arg*. *factor-pair* is a pair of numbers representing the scaling-factor in the X and Y axes. Negative values may be used to produce mirror images.

```
\markup {
  \line {
    \scale #'(2 . 1)
    stretched
    \scale #'(1 . -1)
    mirrored
  }
}
```



`\triangle` *filled* (boolean)
A triangle, either filled or empty.

```
\markup {
  \triangle ##t
  \hspace #2
  \triangle ##f
}
```



Used properties:

- `baseline-skip` (2)
- `font-size` (0)
- `thickness` (0.1)

`\with-url` *url* (string) *arg* (markup)
Add a link to URL *url* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-url #"http://lilypond.org/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

A.10.4 Music

`\customTabClef` *num-strings* (integer) *staff-space* (number)
Draw a tab clef sans-serif style.

`\doubleflat`
Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```



`\doublesharp`
Draw a double sharp symbol.

```
\markup {
  \doublesharp
}
```

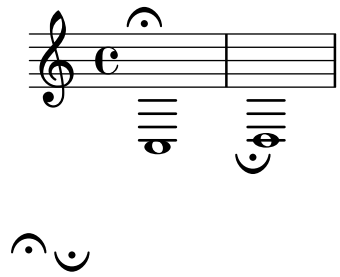


`\fermata`

Create a fermata glyph. When *direction* is `DOWN`, use an inverted glyph. Note that within music, one would usually use the `\fermata` articulation instead of a markup.

```
{ c1^\markup \fermata d1_\markup \fermata }

\markup { \fermata \override #`(direction . ,DOWN) \fermata }
```



Used properties:

- `direction` (1)

`\flat`

Draw a flat symbol.

```
\markup {
  \flat
}
```



`\musicglyph` *glyph-name* (string)

glyph-name is converted to a musical symbol; for example, `\musicglyph #"accidentals.natural"` selects the natural sign from the music font. See Sezione "The Feta font" in *Guida alla Notazione* for a complete listing of the possible glyphs.

```
\markup {
  \musicglyph #"f"
  \musicglyph #"rests.2"
  \musicglyph #"clefs.G_change"
}
```



`\natural`

Draw a natural symbol.

```
\markup {
  \natural
}
```



`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)

Construct a note symbol, with stem and flag. By using fractional values for *dir*, longer or shorter stems can be obtained. Supports all note-head-styles. Supported flag-styles are `default`, `old-straight-flag`, `modern-straight-flag` and `flat-flag`.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```



Used properties:

- `style` ('())
- `flag-style` ('())
- `font-size` (0)

`\note` *duration* (string) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note #"4." #-0.75` creates a dotted quarter note, with a shortened down stem.

```
\markup {
  \override #'(style . cross) {
    \note #"4.." #UP
  }
  \hspace #2
  \note #"breve" #0
}
```



Used properties:

- `style` ('())
- `flag-style` ('())
- `font-size` (0)

`\rest-by-number` *log* (number) *dot-count* (number)

A rest or multi-measure-rest symbol.

```
\markup {
  \rest-by-number #3 #2
  \hspace #2
  \rest-by-number #0 #1
  \hspace #2
  \override #'(multi-measure-rest . #t)
  \rest-by-number #0 #0
}
```



Used properties:

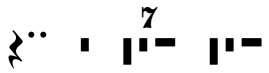
- `multi-measure-rest` (#f)
- `style` ('())
- `font-size` (0)

`\rest` *duration* (string)

This produces a rest, with the *duration* for the rest type and augmentation dots. `"breve"`, `"longa"` and `"maxima"` are valid input-strings.

Printing MultiMeasureRests could be enabled with `\override #'(multi-measure-rest . #t)` If MultiMeasureRests are taken, the MultiMeasureRestNumber is printed above. This is enabled for all styles using default-glyphs. Could be disabled with `\override #'(multi-measure-rest-number . #f)`

```
\markup {
  \rest #"4.."
  \hspace #2
  \rest #"breve"
  \hspace #2
  \override #'(multi-measure-rest . #t)
  {
  \rest #"7"
  \hspace #2
  \override #'(multi-measure-rest-number . #f)
  \rest #"7"
  }
}
```



Used properties:

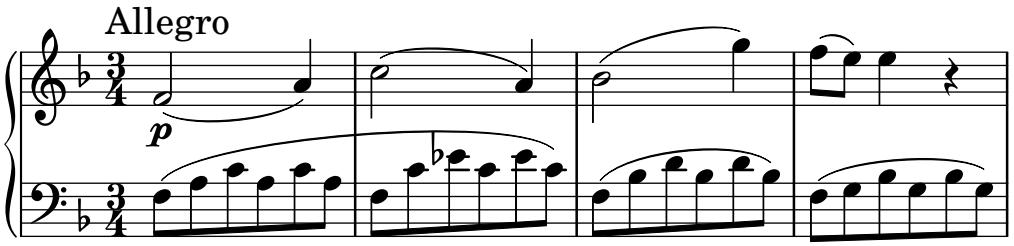
- `word-space` (0.6)
- `multi-measure-rest-number` (#t)
- `multi-measure-rest` (#f)
- `style` ('())

`\score` *score* (score)

Inline an image of music. The reference point (usually the middle staff line) of the lowest staff in the top system is placed on the baseline.

```
\markup {
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
        c2( a4)
        bes2( g'4)
        f8( e) e4 r
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
        f c' es c es c)
```

```
            f,( bes d bes d bes)
            f( g bes g bes g)
          }
        >>
        \layout {
          indent = 0.0\cm
          \context {
            \Score
            \override RehearsalMark
              #'break-align-symbols = #'(time-signature key-signature)
            \override RehearsalMark
              #'self-alignment-X = #LEFT
          }
          \context {
            \Staff
            \override TimeSignature
              #'break-align-anchor-alignment = #LEFT
          }
        }
      }
    }
```



Used properties:

- `baseline-skip`

`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```



`\semisharp`

Draw a semisharp symbol.

```
\markup {
  \semisharp
}
```



`\sesquiflat`

Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```



`\sesquisharp`
: Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```



`\sharp`
: Draw a sharp symbol.

```
\markup {
  \sharp
}
```



`\tied-lyric` *str* (string)
: Like simple-markup, but use tie characters for '~' tilde symbols.

```
\markup \column {
  \tied-lyric #"Siam navi~all'onde~algenti Lasciate~in abbandono"
  \tied-lyric #"Impetuosi venti I nostri~affetti sono"
  \tied-lyric #"Ogni diletto~e scoglio Tutta la vita~e~un mar."
}
```

Siam navi‿all'onde‿algenti Lasciate‿in abbandono
Impetuosi venti I nostri‿affetti sono
Ogni diletto‿e scoglio Tutta la vita‿e‿un mar.

Used properties:

- `word-space`

A.10.5 Instrument Specific Markup

`\fret-diagram` *definition-string* (string)
: Make a (guitar) fret diagram. For example, say

`\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"`

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
  - `s:`*number* – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - `t:`*number* – Set the line thickness (relative to normal line thickness). Default: 0.5.

- `h:`*number* – Set the height of the diagram in frets. Default: 4.
- `w:`*number* – Set the width of the diagram in strings. Default: 6.
- `f:`*number* – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
- `d:`*number* – Set radius of dot, in terms of fret spacing. Default: 0.25.
- `p:`*number* – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
- `c:`*string1*`-`*string2*`-`*fret* – Include a barre mark from *string1* to *string2* on *fret*.
- *string*`-`*fret* – Place a dot on *string* at *fret*. If *fret* is '`o`', *string* is identified as open. If *fret* is '`x`', *string* is identified as muted.
- *string*`-`*fret*`-`*fingering* – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.

– Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by '`x`'.
- Open strings are indicated by '`o`'.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a `-`, followed by the finger indicator, e.g. '`3-2`' for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-verbose` *marking-list* (pair)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
  #'((mute 6) (mute 5) (open 4)
     (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute` *`string-number`*`)`
: Place a small 'x' at the top of string *string-number*.

`(open` *`string-number`*`)`
: Place a small 'o' at the top of string *string-number*.

`(barre` *`start-string end-string fret-number`*`)`
: Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(capo` *`fret-number`*`)`
: Place a capo indicator (a large solid bar) across the entire fretboard at fret location *fret-number*. Also, set fret *fret-number* to be the lowest fret on the fret diagram.

`(place-fret` *`string-number fret-number`* `[`*`finger-value`*` [`*`color-modifier`*`]])`
: Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*, and an optional color modifier *color-modifier*. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the variable *dot-color*. Setting *color-modifier* to `inverted` inverts the dot color for a specific fingering. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\harp-pedal` *definition-string* (string)
: Make a harp pedal diagram.

  Possible elements in *definition-string*:

  `^` pedal is up

  `-` pedal is neutral

  `v` pedal is down

  `|` vertical divider line

  `o` the following pedal should be circled (indicating a change)

  The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you'll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (`\override Voice.TextScript #'size = #0.3`) for the overall, the thickness property (`\override Voice.TextScript #'thickness = #3`) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the harp-pedal-details list of properties (`\override Voice.TextScript #'harp-pedal-details #'box-width = #1`). It contains the following settings: `box-offset` (vertical shift of the box center for up/down pedals), `box-width`, `box-height`, `space-before-divider` (the spacing between two boxes before the divider) and `space-after-divider` (box spacing after the divider).

```
\markup \harp-pedal #"^-v|--ov^"
```



Used properties:

- `thickness` (0.5)
- `harp-pedal-details` ('())
- `size` (1.2)

`\woodwind-diagram` *instrument* (symbol) *user-draw-commands* (list)
Make a woodwind-instrument diagram. For example, say

```
\markup \woodwind-diagram
  #'oboe #'((lh . (d ees)) (cc . (five3qT1q)) (rh . (gis)))
```

for an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed while the five-hole of the central column effectuates a trill between 1/4 and 3/4 closed.

The following instruments are supported:

– piccolo
– flute
– oboe
– clarinet
– bass-clarinet
– saxophone
– bassoon
– contrabassoon

To see all of the callable keys for a given instrument, include the function `(print-keys 'instrument)` in your .ly file, where instrument is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

– 1q (1/4 covered)
– 1h (1/2 covered)
– 3q (3/4 covered)
– R (ring depressed)
– F (fully covered; the default if no state put)

Additionally, these configurations can be used in trills. So, for example, `three3qTR` effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, `threeRT` effectuates a trill between R and open, whereas `threeTR` effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke `(print-keys-verbose 'instrument)`.

Lastly, substituting an empty list for the pressed-key alist will result in a diagram with all of the keys drawn but none filled, for example:

```
\markup \woodwind-diagram #'oboe #'()
```

Used properties:

- `graphical` (#t)
- `thickness` (0.1)
- `size` (1)

A.10.6 Accordion Registers

`\discant` *name* (string)

`\discant` *name* generates a discant accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The register names in the default `\discant` register set have modeled after numeric Swiss notation like depicted in http://de.wikipedia.org/wiki/Register_%28Akkordeon%29, omitting the slashes and dropping leading zeros.

The string *name* is basically a three-digit number with the lowest digit specifying the number of 16' reeds, the tens the number of 8' reeds, and the hundreds specifying the number of 4' reeds. Without modification, the specified number of reeds in 8' is centered in the symbol. Newer instruments may have registrations where 8' can be used either within or without a tone chamber, 'cassotto'. Notationally, the central dot then indicates use of cassotto. One can suffix the tens' digits '`1`' and '`2`' with '`+`' or '`-`' to indicate clustering the dots at the right or left respectively rather than centered.

Some examples are



\discant #"1"



\discant #"1+0"



\discant #"120"



\discant #"131"

Used properties:

- `font-size` (0)

`\freeBass` *name* (string)

`\freeBass` *name* generates a free bass/converter accordion register symbol for the usual two-reed layout.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

Available registrations are



\freeBass #"1"



\freeBass #"11"



\freeBass #"10"

Used properties:

- `font-size` (0)

`\stdBass` *name* (string)

`\stdBass` ***name*** generates a standard bass accordion register symbol.

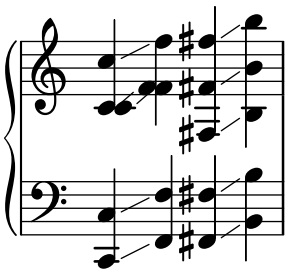
To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The default bass register definitions have been modeled after the article http://www.accordions.com/index/art/stradella.shtml originally appearing in Accord Magazine.

The underlying register model is



This kind of overlapping arrangement is common for Italian instruments though the exact location of the octave breaks differ.

When not composing for a particular target instrument, using the five reed definitions makes more sense than using a four reed layout: in that manner, the '`Master`' register is unambiguous. This is rather the rule in literature bothering about bass registrations at all.

Available registrations are



\stdBass #"Soprano"



\stdBass #"Soft Bass"



\stdBass #"Alto"



\stdBass #"Soft Tenor"



\stdBass #"Tenor"



\stdBass #"Bass/Alto"



\stdBass #"Master"

Used properties:

- `font-size` (0)

`\stdBassIV` *name* (string)
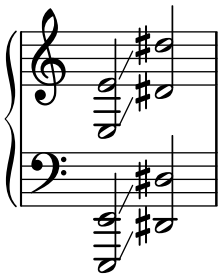
`\stdBassIV` ***name*** generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The main use is for four-reed standard bass instruments with reedbank layout



Notable instruments are Morino models with MIII (the others are five-reed instead) and the Atlantic IV. Most of those models have three register switches. Some newer Morinos with MIII might have five or even seven.

The prevalent three-register layout uses the middle three switches '`Tenor`', '`Master`', '`Soft Bass`'. Note that the sound is quite darker than the same registrations of '`c,`'-based instruments.

Available registrations are



\stdBassIV #"Soprano"



\stdBassIV #"Soft Bass"



\stdBassIV #"Alto"



\stdBassIV #"Bass/Alto"



\stdBassIV #"Tenor"



\stdBassIV #"Soft Bass/Alto"



\stdBassIV #"Master"



\stdBassIV #"Soft Tenor"

Used properties:

- `font-size` (0)

`\stdBassV` *name* (string)
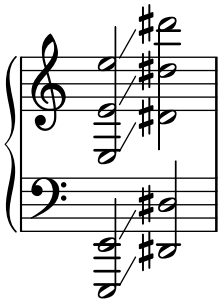
`\stdBassV` ***name*** generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The main use is for five-reed standard bass instruments with reedbank layout



This tends to be the bass layout for Hohner's Morino series without convertor or MIII manual.

With the exception of the rather new 7-register layout, the highest two chord reeds are usually sounded together. The Older instruments offer 5 or 3 bass registers. The Tango VM offers an additional '`Solo Bass`' setting that mutes the chord reeds. The symbol on the register buttons of the Tango VM would actually match the physical five-octave layout reflected here, but it is not used in literature.

Composers should likely prefer the five-reed versions of these symbols. The mismatch of a four-reed instrument with five-reed symbols is easier to resolve for the player than the other way round.

Available registrations are



\stdBassV #"Bass/Alto"



\stdBassV #"Soft Bass"



\stdBassV #"Soft Bass/Alto"



\stdBassV #"Soft Tenor"



\stdBassV #"Alto"



\stdBassV #"Soprano"



\stdBassV #"Tenor"



\stdBassV #"Sopranos"



\stdBassV #"Master"



\stdBassV #"Solo Bass"

Used properties:

- `font-size` (0)

`\stdBassVI` *name* (string)
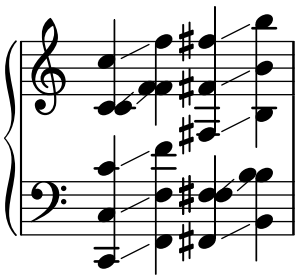
`\stdBassVI` ***name*** generates a standard bass accordion register symbol for six reed basses.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

This is primarily the register layout for the Hohner "Gola" model. The layout is



The registers are effectively quite similar to that of `\stdBass`. An additional bass reed at alto pitch is omitted for esthetical reasons from the '`Master`' setting, so the symbols are almost the same except for the '`Alto/Soprano`' register with bass notes at Alto pitch and chords at Soprano pitch.

Available registrations are



\stdBassVI #"Soprano"



\stdBassVI #"Alto/Soprano"



\stdBassVI #"Alto"



\stdBassVI #"Bass/Alto"



\stdBassVI #"Soft Tenor"



\stdBassVI #"Soft Bass"



\stdBassVI #"Master"

Used properties:

- `font-size` (0)

A.10.7 Other

`\auto-footnote` *mkup* (markup) *note* (markup)

Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```



The footnote will be annotated automatically.

Used properties:

- `padding` (0.0)
- `raise` (0.5)

`\backslashed-digit` *num* (integer)

A feta number, with backslash. This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
```

```
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```

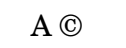


Used properties:

- `thickness` (1.6)
- `font-size` (0)

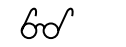
`\char` *num* (integer)
: Produce a single character. Characters encoded in hexadecimal format require the prefix `#x`.

```
\markup {
  \char #65 \char ##x00a9
}
```



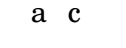
`\eyeglasses`
: Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```



`\footnote` *mkup* (markup) *note* (markup)
: Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```

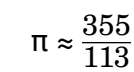


The footnote will not be annotated automatically.

`\fraction` *arg1* (markup) *arg2* (markup)
: Make a fraction of two markups.

```
\markup {

  \fraction 355 113
}
```



Used properties:

- `font-size` (0)

`\fromproperty` *symbol* (symbol)
: Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

```
\header {
  myTitle = "myTitle"
  title = \markup {
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
  \null
}
```

from *myTitle*

`\left-brace` *size* (number)
: A feta brace in point size *size*.

```
\markup {
  \left-brace #35
  \hspace #2
  \left-brace #45
}
```



`\lookup` *glyph-name* (string)
: Lookup a glyph by name.

```
\markup {
  \override #'(font-encoding . fetaBraces) {
    \lookup #"brace200"
    \hspace #2
    \rotate #180
    \lookup #"brace180"
  }
}
```



`\markalphabet` *num* (integer)
: Make a markup letter for *num*. The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

I AA

`\markletter` *num* (integer)
: Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

J AB

`\null`
: An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (procedure) *arg* (markup)
: Apply the *procedure* markup command to *arg*. *procedure* should take a single argument.

`\override` *new-prop* (pair) *arg* (markup)
: Add the argument *new-prop* to the property list. Properties may be any property supported by Sezione “font-interface” in *Guida al Funzionamento Interno*, Sezione “text-interface” in *Guida al Funzionamento Interno* and Sezione “instrument-specific-markup-interface” in *Guida al Funzionamento Interno*.

```
\markup {
  \line {
    \column {
      default
      baseline-skip
    }
    \hspace #2
    \override #'(baseline-skip . 4) {
      \column {
        increased
        baseline-skip
      }
    }
  }
}
```

default baseline-skip
increased baseline-skip

`\page-link` *page-number* (number) *arg* (markup)
Add a link to the page *page-number* around *arg*. This only works in the PDF backend.

```
\markup {
  \page-link #2  { \italic { This links to page 2... } }
}
```

This links to page 2...

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)
Reference to a page number. *label* is the label set on the referenced page (using the `\label` command), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

`\pattern` *count* (integer) *axis* (integer) *space* (number) *pattern* (markup)
Prints *count* times a *pattern* markup. Patterns are spaced apart by *space*. Patterns are distributed on *axis*.

```
\markup \column {
  "Horizontally repeated :"
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated :"
  \pattern #3 #Y #0.5 \flat
}
```

Horizontally repeated :
♭ ♭ ♭ ♭ ♭ ♭ ♭

Vertically repeated :
♭
♭
♭

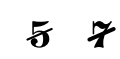
`\property-recursive` *symbol* (symbol)
Print out a warning when a header field markup contains some recursive markup definition.

`\right-brace` *size* (number)
A feta brace in point size *size*, rotated 180 degrees.

```
\markup {
  \right-brace #45
  \hspace #2
  \right-brace #35
}
```

`\slashed-digit` *num* (integer)
A feta number, with slash. This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \slashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\stencil` *stil* (stencil)
: Use a stencil as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```



`\strut`
: Create a box of the same height as the space in the current font.

`\transparent` *arg* (markup)
: Make *arg* transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

`\verbatim-file` *name* (string)
: Read the contents of file *name*, and include it verbatim.

```
\markup {
  \verbatim-file #"simple.ly"
}
```

```
%% A simple piece in LilyPond, a scale.
\relative c' {
  c d e f g a b c
}
%% Optional helper for automatic updating by convert-ly.
%% May be omitted.
\version "2.16.0"
```

`\whiteout` *arg* (markup)
: Provide a white background for *arg*.

```
\markup {
  \combine
    \filled-box #'(-1 . 10) #'(-3 . 4) #1
    \whiteout whiteout
}
```



`\with-color` *color* (color) *arg* (markup)
: Draw *arg* in color specified by *color*.

```
\markup {
  \with-color #red
  red
  \hspace #2
  \with-color #green
  green
  \hspace #2
  \with-color #blue
  blue
}
```

red green blue

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
: Set the dimensions of *arg* to *x* and *y*.

`\with-link` *label* (symbol) *arg* (markup)
: Add a link to the page holding label *label* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-link #'label {
    \italic { This links to the page containing the label... }
  }
}
```

This links to the page containing the label...

A.11 Text markup list commands

The following commands can all be used with `\markuplist`:

`\column-lines` *args* (markup list)
: Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.

Used properties:

- `baseline-skip`

`\justified-lines` *args* (markup list)
: Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\map-markup-commands` *compose* (procedure) *args* (markup list)
: This applies the function *compose* to every markup in *args* (including elements of markup list command calls) in order to produce a new markup list. Since the return value from a markup list command call is not a markup list but rather a list of stencils, this requires passing those stencils off as the results of individual markup calls. That way, the results should work out as long as no markups rely on side effects.

`\override-lines` *new-prop* (pair) *args* (markup list)
: Like `\override`, for markup lists.

`\table-of-contents`

`\wordwrap-internal` *justify* (boolean) *args* (markup list)
: Internal markup list command used to define `\justify` and `\wordwrap`.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)

`\wordwrap-lines` *args* (markup list)
: Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\wordwrap-string-internal` *justify* (boolean) *arg* (string)
: Internal markup list command used to define `\justify-string` and `\wordwrap-string`.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width`

A.12 List of special characters

The following special characters references can be used; for more details, see [ASCII aliases], pagina 488.

The HTML syntax is used and most of these references are the same as HTML. The rest of them are inspired by LaTeX.

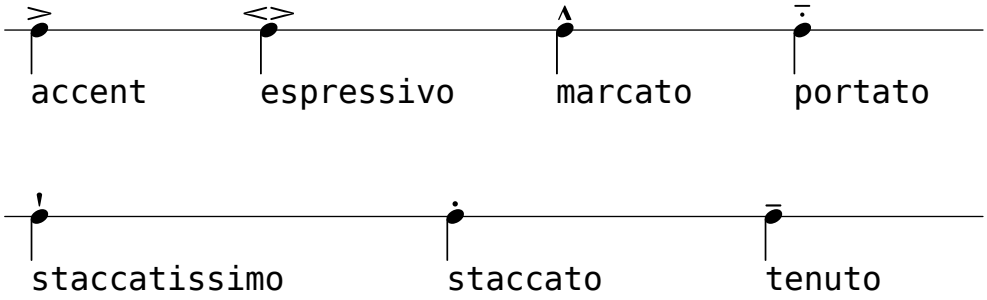
The characters are boxed so that you can see their size. A small padding has been added between the character and the box for more readability.

…	…	–	–	—	—	¡	¡		
¿	¿	&solidus;	⁄	&flq;	‹	&frq;	›		
&flqq;	«	&frqq;	»	&glq;	‚	&grq;	‘		
&glqq;	„	&grqq;	“	&elq;	‘	&erq;	’		
&elqq;	“	&erqq;	”						
		&nnbsp;	‍						
‌	·	·	•	•	&copyright;	©			
&registered;	®	&trademark;	™	†	†	‡	‡		
№	№	ª	ª	º	º	¶	¶		
§	§	°	°	№	№	‰	‰		
¦	¦	´	´	&acutedbl;	˝	`	`		
˘	˘	ˇ	ˇ	&cedilla;	¸	&circumflex;	^		
&diaeresis;	¨	&macron;	¯	&aa;	å	&AA;	Å		
&ae;	æ	&AE;	Æ	&dh;	ð	&DH;	Ð		
&dj;	đ	&DJ;	Đ	&l;	ł	&L;	Ł		
&ng;	ŋ	&NG;	Ŋ	&o;	ø	&O;	Ø		
&oe;	œ	&OE;	Œ	&s;	ß	&ss;	ß		
&th;	þ	&TH;	Þ	+	+	−	−		
×	×	÷	÷	¹	1	²	2		
³	3	&sqrt;	√	&increment;	Δ	&infty;	∞		
∑	∑	±	±	&bulletop;	∙	&partial;	∂		
&neg;	¬	&currency;	¤	$	$	€	€		
&pounds;	£	¥	¥	¢	¢				

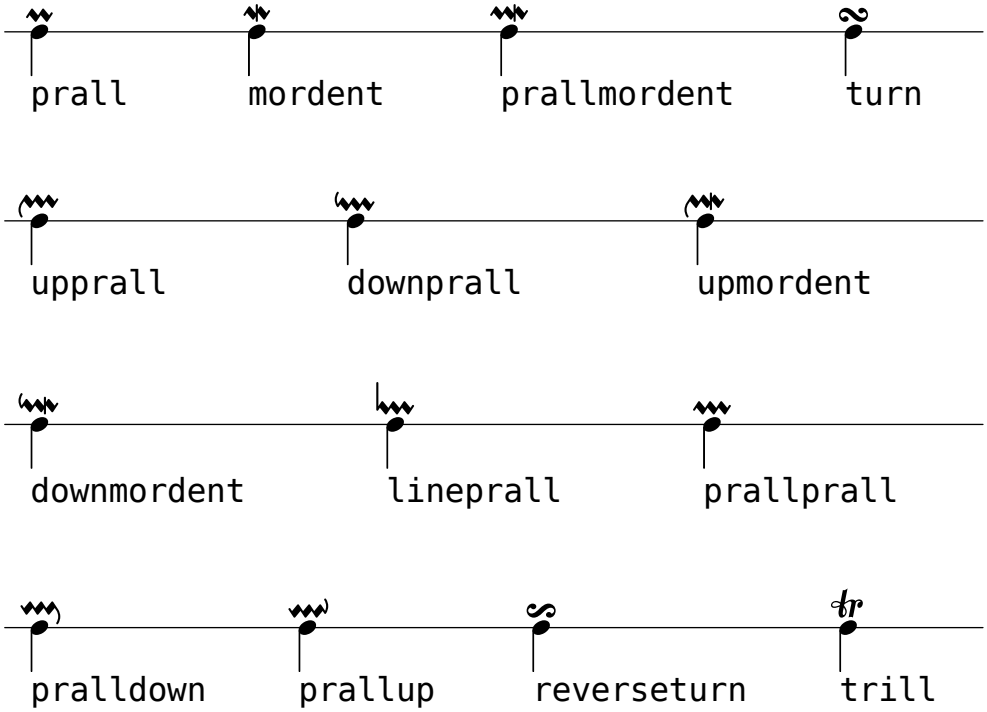
A.13 List of articulations

The following scripts are available in the Feta font and may be attached to notes (eg. '`c\accent`').

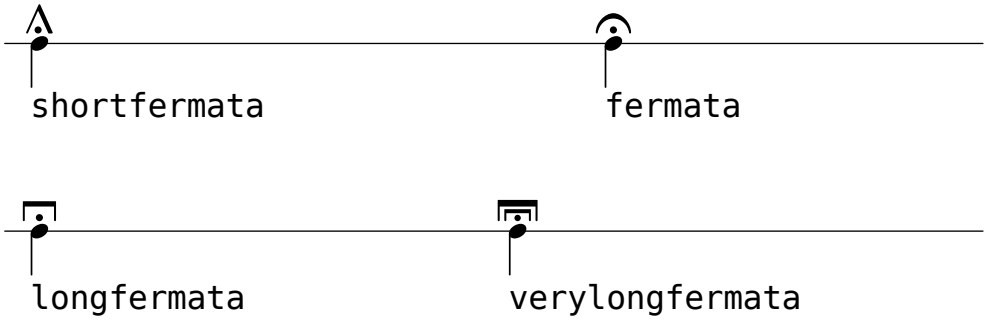
Articulation scripts



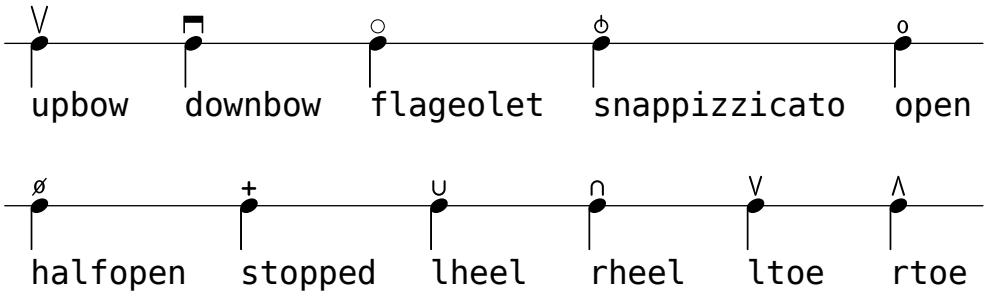
Ornament scripts



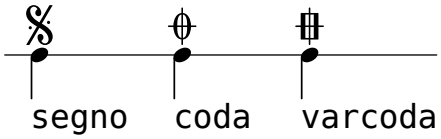
Fermata scripts



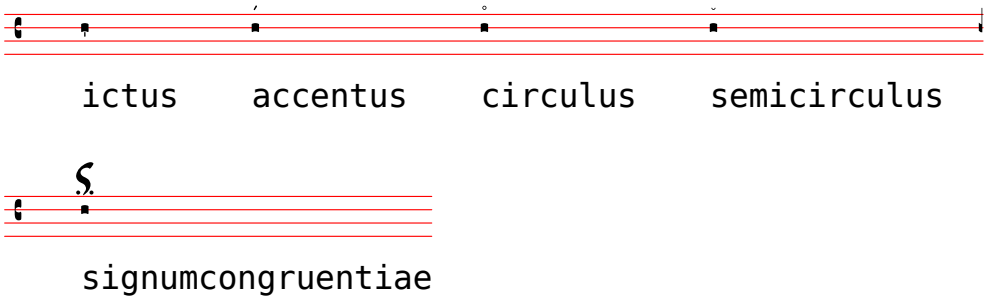
Instrument-specific scripts



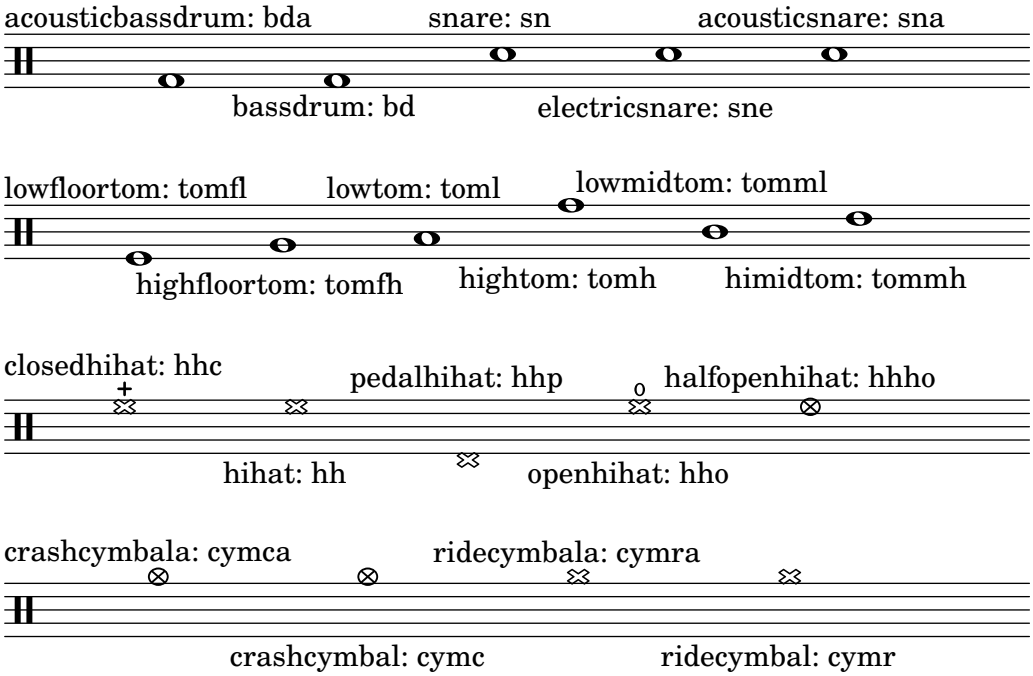
Repeat sign scripts

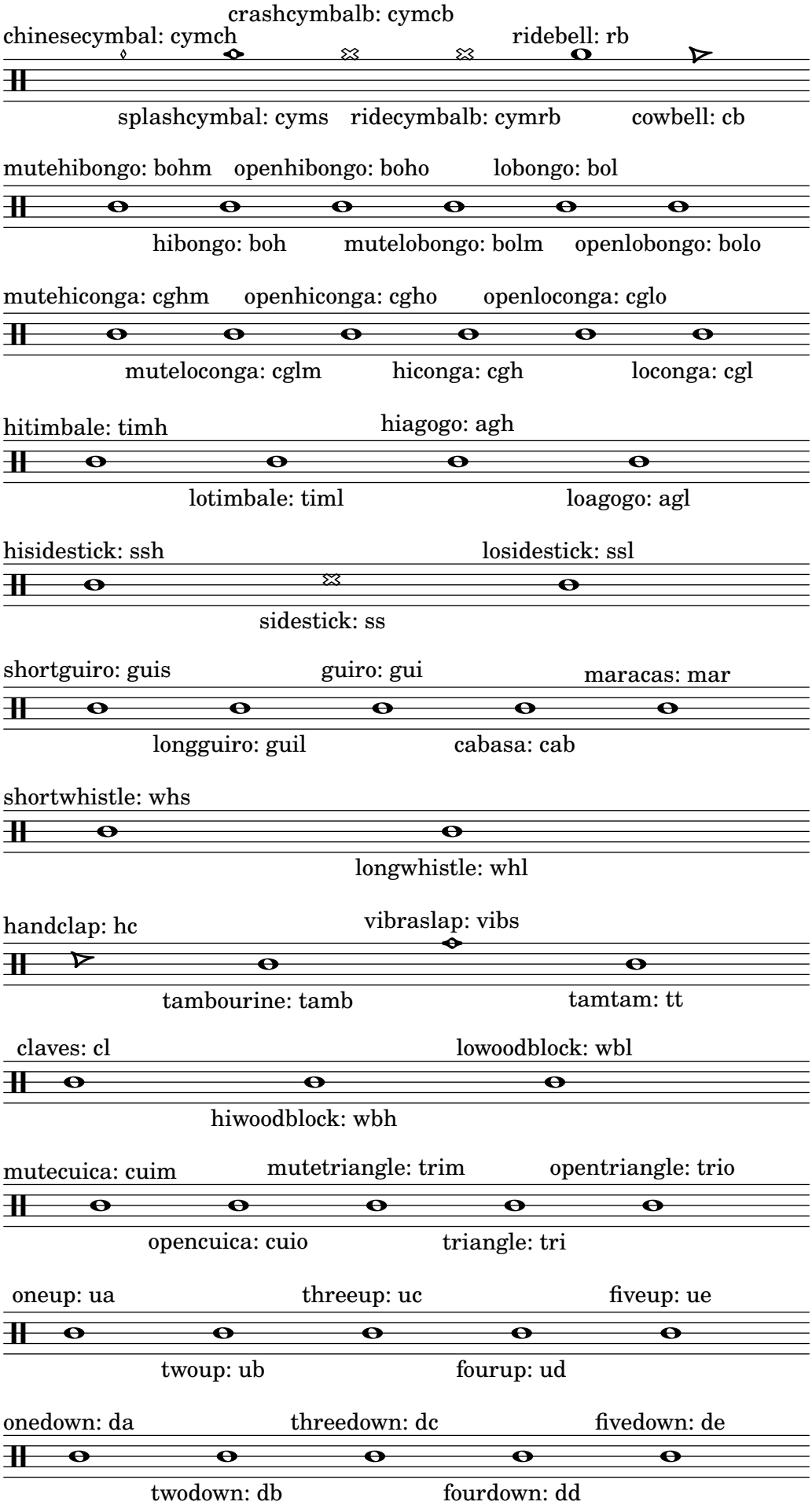


Ancient scripts



A.14 Percussion notes




crashcymbalb: cymcb
chinesecymbal: cymch
ridebell: rb
splashcymbal: cyms
ridecymbalb: cymrb
cowbell: cb
mutehibongo: bohm
openhibongo: boho
lobongo: bol
hibongo: boh
mutelobongo: bolm
openlobongo: bolo
mutehiconga: cghm
openhiconga: cgho
openloconga: cglo
muteloconga: cglm
hiconga: cgh
loconga: cgl
hitimbale: timh
hiagogo: agh
lotimbale: timl
loagogo: agl
hisidestick: ssh
losidestick: ssl
sidestick: ss
shortguiro: guis
guiro: gui
maracas: mar
longguiro: guil
cabasa: cab
shortwhistle: whs
longwhistle: whl
handclap: hc
vibraslap: vibs
tambourine: tamb
tamtam: tt
claves: cl
lowoodblock: wbl
hiwoodblock: wbh
mutecuica: cuim
mutetriangle: trim
opentriangle: trio
opencuica: cuio
triangle: tri
oneup: ua
threeup: uc
fiveup: ue
twoup: ub
fourup: ud
onedown: da
threedown: dc
fivedown: de
twodown: db
fourdown: dd

A.15 Technical glossary

A glossary of the technical terms and concepts used internally in LilyPond. These terms may appear in the manuals, on mailing lists or in the source code.

alist

An association list or **alist** for short is a Scheme pair which associates a value with a key: `(key . value)`. For example, in 'scm/lily.scm', the alist "type-p-name-alist" associates certain type predicates (e.g. `ly:music?`) with names (e.g. "music") so that type-check failures can be reported with a console message that includes the name of the expected type predicate.

callback

A **callback** is a routine, function or method whose reference is passed as an argument in a call to another routine, so allowing the called routine to invoke it. The technique enables a lower-level software layer to call a function defined in a higher layer. Callbacks are used extensively in LilyPond to permit user-level Scheme code to define how many low-level actions are performed.

closure

In Scheme, a **closure** is created when a function, usually a lambda expression, is passed as a variable. The closure contains the function's code plus references to the lexical bindings of the function's free variables (i.e. those variables used in the expression but defined outside it). When this function is applied to different arguments later, the free variable bindings that were captured in the closure are used to obtain the values of the free variables to be used in the calculation. One useful property of closures is the retention of internal variable values between invocations, so permitting state to be maintained.

A **simple closure** is a closure whose expression has no free variables and hence no free variable bindings.

A simple closure is represented in LilyPond by a smob containing the expression and a method to apply the expression to a passed list of arguments.

glyph

A **glyph** is a particular graphical representation of a typographic character, or a combination of two characters formating a ligature. A set of glyphs with a single style and shape comprise a font, and a set of fonts covering several styles and sizes comprise a typeface.

Vedi anche

Notation Reference: ⟨undefined⟩ [Fonts], pagina ⟨undefined⟩, Sezione 3.3.3 [Special characters], pagina 486.

grob

LilyPond objects which represent items of notation in the printed output such as note heads, stems, slurs, ties, fingering, clefs, etc are called 'Layout objects', often known as 'GRaphical OBjects', or **grobs** for short. They are represented by instances of the `Grob` class.

Vedi anche

Learning Manual: Sezione "Objects and interfaces" in *Manuale di Apprendimento*, Sezione "Naming conventions of objects and properties" in *Manuale di Apprendimento*, Sezione "Properties of layout objects" in *Manuale di Apprendimento*.

Internals Reference: Sezione "grob-interface" in *Guida al Funzionamento Interno*, Sezione "All layout objects" in *Guida al Funzionamento Interno*.

immutable

An **immutable** object is one whose state cannot be modified after creation, in contrast to a mutable object, which can be modified after creation.

In LilyPond, immutable or shared properties define the default style and behavior of grobs. They are shared between many objects. In apparent contradiction to the name, they can be changed using `\override` and `\revert`.

Vedi anche

Notation Reference: [mutable], pagina 710.

interface

Actions and properties which are common to a number of grobs are grouped together in an object called a `grob-interface`, or just 'interface' for short.

Vedi anche

Learning Manual: Sezione "Objects and interfaces" in *Manuale di Apprendimento*, Sezione "Naming conventions of objects and properties" in *Manuale di Apprendimento*, Sezione "Properties found in interfaces" in *Manuale di Apprendimento*.

Notation Reference: Sezione 5.2.2 [Layout interfaces], pagina 572.

Internals Reference: Sezione "Graphical Object Interfaces" in *Guida al Funzionamento Interno*.

lexer

A **lexer** is a program which converts a sequence of characters into a sequence of tokens, a process called lexical analysis. The LilyPond lexer converts the stream obtained from an input '`.ly`' file into a tokenized stream more suited to the next stage of processing - parsing, for which see [parser], pagina 710. The LilyPond lexer is built with Flex from the lexer file '`lily/lexer.ll`' which contains the lexical rules. This file is part of the source code and is not included in the LilyPond binary installation.

mutable

A **mutable** object is one whose state can be modified after creation, in contrast to an immutable object, whose state is fixed at the time of creation.

In LilyPond, mutable properties contain values that are specific to one grob. Typically, lists of other objects or results from computations are stored in mutable properties.

Vedi anche

Notation Reference: [immutable], pagina 710.

output-def

An instance of the `Output-def` class contains the methods and data structures associated with an output block. Instances are created for midi, layout and paper blocks.

parser

A **parser** analyzes the sequence of tokens produced by a lexer to determine its grammatical structure, grouping the tokens progressively into larger groupings according to the rules of the grammar. If the sequence of tokens is valid the end product is a tree of tokens whose root is the grammar's start symbol. If this cannot be achieved the file is invalid and an appropriate error message is produced. The syntactic groupings and the rules for constructing the groupings

from their parts for the LilyPond syntax are defined in '`lily/parser.yy`' and shown in Backus Normal Form (BNF) in Sezione "LilyPond grammar" in *Guida del Collaboratore*. This file is used to build the parser during the program build by the parser generator, Bison. It is part of the source code and is not included in the LilyPond binary installation.

parser variable

These are variables defined directly in Scheme. Their direct use by users is strongly discouraged, because their scoping semantics can be confusing.

When the value of such a variable is changed in a '`.ly`' file, the change is global, and unless explicitly reverted, the new value will persist to the end of the file, affecting subsequent `\score` blocks as well as external files added with the `\include` command. This can lead to unintended consequences and in complex typesetting projects the consequent errors can be difficult to track down.

LilyPond uses the following parser variables:

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- partCombineListener
- pitchnames
- toplevel-bookparts
- toplevel-scores
- showLastLength
- showFirstLength

prob

PRoperty OBjects, or **probs** for short, are instances of the `Prob` class, a simple base class for objects which have mutable and immutable property alists and the methods to manipulate them. The `Music` and `Stream_event` classes derive from `Prob`. Instances of the `Prob` class are also created to hold the formatted content of system grobs and titling blocks during page layout.

simple closure

See [closure], pagina 709.

smob

Smobs, or ScheMe OBjects, are part of the mechanism used by Guile to export C and C++ objects to Scheme code. In LilyPond, smobs are created from C++ objects through macros. There are two types of smob objects: simple smobs, intended for simple immutable objects like numbers, and complex smobs, used for objects with identities. If you have access to the LilyPond sources, more information can be found in '`lily/includes/smob.hh`'.

stencil

An instance of the **stencil** class holds the information required to print a typographical object. It is a simple smob containing a confining box, which defines the vertical and horizontal extents of the object, and a Scheme expression which will print the object when evaluated. Stencils may be combined to form more complex stencils defined by a tree of Scheme expressions formed from the Scheme expressions of the component stencils.

The `stencil` property, which connects a grob to its stencil, is defined in the `grob-interface` interface.

Vedi anche

Internals Reference: Sezione “grob-interface” in *Guida al Funzionamento Interno*.

A.16 All context properties

`accidentalGrouping` (symbol)
: If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`additionalPitchPrefix` (string)
: Text with which to prefix additional pitches within a chord name.

`aDueText` (markup)
: Text to print at a unisono passage.

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBassFigureAccidentals` (boolean)
: If true, then the accidentals are aligned in bass figure context.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`alternativeNumberingStyle` (symbol)
: The style of an alternative’s bar numbers. Can be `numbers` for going back to the same number or `numbers-with-letters` for going back to the same number with letter suffixes. No setting will not go back in measure-number time.

`associatedVoice` (string)
: Name of the `Voice` that has the melody for this `Lyrics` line.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Sezione “Score” in *Guida al Funzionamento Interno* then all staves share accidentals, and if *context* is Sezione “Staff” in *Guida al Funzionamento Interno* then all voices in the same staff share accidentals, but staves do not.

  procedure The procedure represents an accidental rule to be applied to the previously specified context.

  The procedure takes the following arguments:

  `context` The current context to which the rule should be applied.

  `pitch` The pitch of the note to be evaluated.

  `barnum` The current bar number.

  `measurepos` The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoBeamCheck` (procedure)
: A procedure taking three arguments, *context*, *dir* [start/stop (-1 or 1)], and *test* [shortest note in the beam]. A non-`#f` return value starts or stops the auto beam.

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`automaticBars` (boolean)
: If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

`barAlways` (boolean)
: If set to true a bar line is drawn after each note.

`barCheckSynchronize` (boolean)
: If true then reset `measurePosition` when finding a bar check.

`barNumberFormatter` (procedure)
: A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

`barNumberVisibility` (procedure)
: A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the `break-visibility` property.

  The following procedures are predefined:

  `all-bar-numbers-visible`
  : Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

  `first-bar-number-invisible`
  : Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

  `first-bar-number-invisible-save-broken-bars`
  : Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

  `first-bar-number-invisible-and-no-parenthesized-bar-numbers`
  : Enable bar numbers for all bars except the first bar and broken bars. This is the default.

  `(every-nth-bar-number-visible` *n*`)`
  : Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

  `(modulo-bar-number-visible` *n m*`)`
  : If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`bassFigureFormatFunction` (procedure)
: A procedure that is called to produce the formatting for a `BassFigure` grob. It takes a list of `BassFigureEvent`s, a context, and the grob to format.

`bassStaffProperties` (list)
: An alist of property settings to apply for the down staff of `PianoStaff`. Used by `\autochange`.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`chordChanges` (boolean)
: Only show changes in chords scheme?

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameExceptionsFull` (list)
: An alist of full chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameExceptionsPartial` (list)
: An alist of partial chord exceptions. Contains `(`*`chord . (prefix-markup suffix-markup`*`))` entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNameLowercaseMinor` (boolean)
: Downcase roots of minor chords?

`chordNameSeparator` (markup)
: The markup object used to separate parts of a chord name.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordPrefixSpacer` (number)
: The space added between the root symbol and the prefix of a chord name.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`completionBusy` (boolean)
: Whether a completion-note head is playing.

`completionUnit` (moment)
: Sub-bar unit of completion.

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

`defaultBarType` (string)
: Set the default type of bar line. See `whichBar` for information on available bar types.

  This variable is read by Sezione "Timing_translator" in *Guida al Funzionamento Interno* at Sezione "Score" in *Guida al Funzionamento Interno* level.

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`doubleRepeatSegnoType` (string)
: Set the default bar line for the combinations double repeat with segno. Default is '`:|.S.|:`'.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`drumPitchTable` (hash table)
: A table mapping percussion instruments (symbols) to pitches.

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`', '`timbales-style`', '`congas-style`', '`bongos-style`', and '`percussion-style`'.

  The layout style is a hash table, containing the drum-pitches (e.g., the symbol '`hihat`') as keys, and a list `(notehead-style script vertical-position)` as values.

`endRepeatSegnoType` (string)
: Set the default bar line for the combinations ending of repeat with segno. Default is '`:|.S`'.

`endRepeatType` (string)
: Set the default bar line for the ending of repeats.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extendersOverRests` (boolean)
: Whether to continue extenders as they cross a rest.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`figuredBassPlusDirection` (direction)
: Where to put plus signs relative to the main figure.

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`firstClef` (boolean)
: If true, create a new clef when starting a staff.

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

`fontSize` (number)
: The relative size of all grobs in a context.

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

`handleNegativeFrets` (symbol)
: How the automatic fret calculator should handle calculated negative frets. Values include `'ignore`, to leave them out of the diagram completely, `'include`, to include them as calculated, and `'recalculate`, to ignore the specified string and find a string where they will fit with a positive fret number.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`ignoreBarChecks` (boolean)
: Ignore bar checks.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`ignoreMelismata` (boolean)
: Ignore melismata for this Sezione “Lyrics” in *Guida al Funzionamento Interno* line.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`implicitTimeSignatureVisibility` (vector)
: break visibility for the default time signature.

`includeGraceNotes` (boolean)
: Do not ignore grace notes for Sezione “Lyrics” in *Guida al Funzionamento Interno*.

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

`instrumentEqualizer` (procedure)
: A function taking a string (instrument name), and returning a `(`*`min . max`*`)` pair of numbers for the loudness range of the instrument.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`instrumentTransposition` (pitch)
: Define the transposition of the instrument. Its value is the pitch that sounds when the instrument plays written middle C. This is used to transpose the MIDI output, and `\quote`s.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step . alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keySignature` (list)
: The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keySignature = #`((6 . ,FLAT))`.

`lyricMelismaAlignment` (number)
: Alignment to use for a melisma syllable.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`markFormatter` (procedure)
: A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`maximumFretStretch` (number)
: Don’t allocate frets further than this from specified frets.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`melismaBusyProperties` (list)
: A list of properties (symbols) to determine whether a melisma is playing. Setting this property will influence how lyrics are aligned to notes. For example, if set to `'(melismaBusy beamMelismaBusy)`, only manual melismata and manual beams are considered. Possible values include `melismaBusy`, `slurMelismaBusy`, `tieMelismaBusy`, and `beamMelismaBusy`.

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`midiBalance` (number)
: Stereo balance for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to leftmost emphasis, center balance, and rightmost emphasis, respectively.

`midiChannelMapping` (symbol)
: How to map MIDI channels: per `staff` (default), `instrument` or `voice`.

`midiChorusLevel` (number)
: Chorus effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMergeUnisons` (boolean)
: If true, output only one MIDI note-on event when notes with the same pitch, in the same MIDI-file track, overlap.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`midiPanPosition` (number)
: Pan position for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to hard left, center, and hard right, respectively.

`midiReverbLevel` (number)
: Reverb effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`minimumPageTurnLength` (moment)
: Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)
: Minimum length of a repeated section for a page turn to be allowed within that section.

`minorChordModifier` (markup)
: Markup displayed following the root for a minor chord

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

`output` (music output)
: The output produced by a score-level translator during music interpretation.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

`predefinedDiagramTable` (hash table)
: The hash table of predefined fret diagrams to use in FretBoards.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

`printOctaveNames` (boolean)
: Print octave marks for the `NoteNames` context.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

`rehearsalMark` (integer)
: The last rehearsal mark printed.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta x)`, where *x* is a string or `#f`. `'end-repeat` is also accepted as a command.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

`restCompletionBusy` (boolean)
: Signal whether a completion-rest is active.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`restrainOpenStrings` (boolean)
: Exclude open strings from the automatic fret calculator.

`searchForVoice` (boolean)
: Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

`segnoType` (string)
: Set the default bar line for a requested segno. Default is '`S`'.

`shapeNoteStyles` (vector)
: Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`skipBars` (boolean)
: If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

```
{
  r1 r1*3 R1*3
  \set Score.skipBars= ##t
  r1*3 R1*3
}
```

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`slashChordSeparator` (markup)
: The markup object used to separate a chord name from its root note in case of inversions or slash chords.

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

`squashedPosition` (integer)
: Vertical position of squashing for Sezione "Pitch_squash_engraver" in *Guida al Funzionamento Interno*.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

`stanza` (markup)
: Stanza 'number' to print before the start of a verse. Use in `Lyrics` context.

`startRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno. Default is '`S.|:`'.

`startRepeatType` (string)
: Set the default bar line for the beginning of repeats.

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`strictBeatBeaming` (boolean)
: Should partial beams reflect the beat structure even if it causes flags to hang out?

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

`suggestAccidentals` (boolean)
: If set, accidentals are typeset as cautionary suggestions over the note.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

`tempoWholesPerMinute` (moment)
: The tempo in whole notes per minute.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

`timeSignatureSettings` (list)
: A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for `baseMoment`, `beatStructure`, and `beamExceptions`.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

`tonic` (pitch)
: The tonic of the current scale.

`topLevelAlignment` (boolean)
: If true, the *Vertical_align_engraver* will create a *VerticalAlignment*; otherwise, it will create a *StaffGrouper*

`trebleStaffProperties` (list)
: An alist of property settings to apply for the up staff of `PianoStaff`. Used by `\autochange`.

`tremoloFlags` (integer)
: The number of tremolo flags to add if no number is specified.

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

`tupletSpannerDuration` (moment)
: Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. By setting this property, you can make brackets last shorter.

```
{
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

`vocalName` (markup)
: Name of a vocal line.

`voltaSpannerDuration` (moment)
: This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in '`scm/bar-line.scm`'.

A.17 Layout properties

`add-stem-support` (boolean)
: If set, the `Stem` object is included in this script's support.

`after-line-breaking` (boolean)
: Dummy property, used to trigger callback for `after-line-breaking`.

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`allow-loose-spacing` (boolean)
: If set, column can be detached from main spacing.

`allow-span-bar` (boolean)
: If false, no inter-staff bar line will be created below this bar line.

`alteration` (number)
: Alteration numbers for accidental.

`alteration-alist` (list)
: List of `(`*`pitch . accidental`*`)` pairs for key signature.

`annotation` (string)
: Annotate a grob for debug purposes.

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean)
: Print the line from an annotation to the grob that it annotates.

`arpeggio-direction` (direction)
: If set, put an arrow on the arpeggio squiggly line.

`arrow-length` (number)
: Arrow length.

`arrow-width` (number)
: Arrow width.

`auto-knee-gap` (dimension, in staff space)
: If a gap is found between note heads where a horizontal beam fits that is larger than this number, make a kneed beam.

`automatically-numbered` (boolean)
: Should a footnote be automatically numbered?

`average-spacing-wishes` (boolean)
: If set, the spacing wishes are averaged over staves.

`avoid-note-head` (boolean)
: If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

`avoid-scripts` (boolean)
: If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`axes` (list)
: List of axis numbers. In the case of alignment grobs, this should contain only one number.

`bar-extent` (pair of numbers)
: The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`base-shortest-duration` (moment)
: Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`baseline-skip` (dimension, in staff space)
: Distance between base lines of multiple lines of text.

`beam-thickness` (dimension, in staff space)
: Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space)
: Width of the tremolo sign.

`beamed-stem-shorten` (list)
: How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair)
: Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`beamlet-default-length` (pair)
: A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair)
: The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`before-line-breaking` (boolean)
: Dummy property, used to trigger a callback function.

`between-cols` (pair)
: Where to attach a loose column to.

`bound-details` (list)
: An alist of properties for determining attachments of spanners to edges.

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`bracket-visibility` (boolean or symbol)
: This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-align-anchor` (number)
: Grobs aligned to this break-align grob will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)
: Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-orders` (vector)
: Defines the order in which prefatory matter (clefs, key signatures) appears. The format is a vector of length 3, where each element is one order for end-of-line, middle of line, and start-of-line, respectively. An order is a list of symbols.

  For example, clefs are put after key signatures by setting

```
\override Score.BreakAlignment #'break-align-orders =
  #(make-vector 3 '(span-bar
                    breathing-sign
                    staff-bar
                    key
                    clef
                    time-signature))
```

`break-align-symbol` (symbol)
: This key is used for aligning and spacing breakable items.

`break-align-symbols` (list)
: A list of symbols that determine which break-aligned grobs to align this to. If the grob selected by the first symbol in the list is invisible due to break-visibility, we will align to the next grob (and so on). Choices are `left-edge`, `ambitus`, `breathing-sign`, `clef`, `staff-bar`, `key-cancellation`, `key-signature`, `time-signature`, and `custos`.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`break-visibility` (vector)
: A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`breakable` (boolean)
: Allow breaks here.

`broken-bound-padding` (number)
: The amount of padding to insert when a spanner is broken at a line break.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`clip-edges` (boolean)
: Allow outward pointing beamlets at the edges of beams?

`collapse-height` (dimension, in staff space)
: Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`collision-bias` (number)
: Number determining how much to favor the left (negative) or right (positive). Larger absolute values in either direction will push a collision in this direction.

`collision-interfaces` (list)
: A list of interfaces for which automatic beam-collision resolution is run.

`collision-padding` (number)
: Amount of padding to apply after a collision is detected via the self-alignment-interface.

`collision-voice-only` (boolean)
: Does automatic beam collsion apply only to the voice in which the beam was created?

`color` (color)
: The color of this grob.

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`concaveness` (number)
: A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`control-points` (list)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`count-from` (integer)
: The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`damping` (number)
: Amount of beam slope damping.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`dash-fraction` (number)
: Size of the dashes, relative to `dash-period`. Should be between `0.0` (no line) and `1.0` (continuous line).

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`default-direction` (direction)
: Direction determined by note head positions.

`default-staff-staff-spacing` (list)
: The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`digit-names` (vector)
: Names for string finger digits.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`dot-count` (integer)
: The number of dots.

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`dot-placement-list` (list)
: List consisting of `(`*`description string-number fret-number finger-number`*`)` entries used to define fret diagrams.

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`eccentricity` (number)
: How asymmetrical to make a slur. Positive means move the center to the right.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`edge-text` (pair)
: A pair specifying the texts to be set at the edges: `(`*`left-text . right-text`*`)`.

`expand-limit` (integer)
: Maximum number of measures expanded in church rests.

`extra-dy` (number)
: Slope glissandi this much extra.

`extra-offset` (pair of numbers)
: A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff's `StaffSymbol`.

`extra-spacing-height` (pair of numbers)
: In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers)
: In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flag-count` (number)
: The number of tremolo beams.

`flat-positions` (list)
: Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`font-encoding` (symbol)
: The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-family` (symbol)
: The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-name` (string)
: Specifies a file name (without extension) of the font to load. This setting overrides selection using `font-family`, `font-series` and `font-shape`.

`font-series` (symbol)
: Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol)
: Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number)
: The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. Fractional values are allowed.

`footnote` (boolean)
: Should this be a footnote or in-note?

`footnote-music` (music)
: Music creating a footnote.

`footnote-text` (markup)
: A footnote for the grob.

`force-hshift` (number)
: This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Sezione "note-collision-interface" in *Guida al Funzionamento Interno*.

`forced-spacing` (number)
: Spacing forced between grobs, used in various ligature engravers.

`fraction` (fraction, as pair)
: Numerator and denominator of a time signature object.

`french-beaming` (boolean)
: Use French beaming style for this stem. The stem stops at the innermost beams.

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a `(`*`property`*` . `*`value`*`)` pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
  - `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
  - `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
  - `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
  - `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
  - `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
  - `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
  - `fret-count` – The number of frets. Default 4.
  - `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
  - `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.
  - `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
  - `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
  - `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
  - `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.
  - `open-string` – Character string to be used to indicate open string. Default `"o"`.

- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`full-length-padding` (number)
: How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)
: Run to the extent of the column for a full-length tuplet bracket.

`full-measure-extra-space` (number)
: Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`full-size-change` (boolean)
: Don't make a change clef smaller.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`gap-count` (integer)
: Number of gapped beams for tremolo.

`glissando-skip` (boolean)
: Should this `NoteHead` be skipped by glissandi?

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

  In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`graphical` (boolean)
: Display in graphical (vs. text) form.

`grow-direction` (direction)
: Crescendo or decrescendo?

`hair-thickness` (number)
: Thickness of the thin line in a bar line.

`harp-pedal-details` (list)
: An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a `(property . value)` pair. The properties which can be included in harp-pedal-details include the following:

  - `box-offset` – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
  - `box-width` – Width of each pedal box. Default value 0.4.
  - `box-height` – Height of each pedal box. Default value 1.0.
  - `space-before-divider` – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
  - `space-after-divider` – Space between boxes after the first divider. Default value 0.8.
  - `circle-thickness` – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
  - `circle-x-padding` – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
  - `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`height-limit` (dimension, in staff space)
: Maximum slur height: The longer the slur, the closer it is to this height.

`hide-tied-accidental-after-break` (boolean)
: If set, an accidental that appears on a tied note after a line break will not be displayed.

`horizon-padding` (number)
: The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`horizontal-shift` (integer)
: An integer that identifies ranking of `NoteColumn`s for horizontal shifting. This is used by Sezione “note-collision-interface” in *Guida al Funzionamento Interno*.

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`id` (string)
: An id string for the grob. Depending on the typestting backend being used, this id will be assigned to a group containing all of the stencils that comprise a given grob. For example, in the svg backend, the string will be assigned to the `id` attribute of a group (`<g>`) that encloses the stencils that comprise the grob. In the Postscript backend, as there is no way to group items, the setting of the id property will have no effect.

`ignore-collision` (boolean)
: If set, don’t do note collision resolution on this `NoteColumn`.

`implicit` (boolean)
: Is this an implicit bass figure?

`inspect-index` (integer)
: If debugging is set, set beam and slur configuration to this index, and print the respective scores.

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur quants to this position, and print the respective scores.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`kern` (dimension, in staff space)
: Amount of extra white space to add. For bar lines, this is the amount of space after a thick line.

`knee` (boolean)
: Is this beam kneed?

`knee-spacing-correction` (number)
: Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`labels` (list)
: List of labels (symbols) placed on a column.

`layer` (integer)
: An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-extra` (dimension, in staff space)
: Extra distance from staff line to draw ledger lines for.

`ledger-line-thickness` (pair of numbers)
: The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`ledger-positions` (list)
: Repeating pattern for the vertical positions of ledger lines. Bracketed groups are always shown together.

`left-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`left-padding` (dimension, in staff space)
: The amount of space that is put left to an object (e.g., a lyric extender).

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems.

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`line-break-penalty` (number)
: Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

`line-break-permission` (symbol)
: Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`line-break-system-details` (list)
: An alist of properties to use if this column is the start of a system.

`line-count` (integer)
: The number of staff lines.

`line-positions` (list)
: Vertical positions of staff lines.

`line-thickness` (number)
: The thickness of the tie or slur contour.

`long-text` (markup)
: Text markup. See Sezione "Formatting text" in *Guida alla Notazione*.

`max-beam-connect` (integer)
: Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

`max-stretch` (number)
: The maximum amount that this `VerticalAxisGroup` can be vertically stretched (for example, in order to better fill a page).

`maximum-gap` (number)
: Maximum value allowed for `gap` property.

`measure-count` (integer)
: The number of measures for a multi-measure rest.

`measure-length` (moment)
: Length of a measure. Used in some spacing situations.

`merge-differently-dotted` (boolean)
: Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

  `merge-differently-dotted` only applies to opposing stem directions (i.e., voice 1 & 2).

`merge-differently-headed` (boolean)
: Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Sezione "note-collision-interface" in *Guida al Funzionamento Interno*.

  `merge-differently-headed` only applies to opposing stem directions (i.e., voice 1 & 2).

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-length-fraction` (number)
: Minimum length of ledger line as fraction of note head size.

`minimum-space` (dimension, in staff space)
: Minimum distance that the victim should move (after padding).

`minimum-X-extent` (pair of numbers)
: Minimum size of an object in X dimension, measured in `staff-space` units.

`minimum-Y-extent` (pair of numbers)
: Minimum size of an object in Y dimension, measured in `staff-space` units.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`neutral-position` (number)
: Position (in half staff spaces) where to flip the direction of custos stem.

`next` (graphical (layout) object)
: Object that is next relation (e.g., the lyric syllable following an extender).

`no-alignment` (boolean)
: If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`no-ledgers` (boolean)
: If set, don't draw ledger lines on this object.

`no-stem-extend` (boolean)
: If set, notes with ledger lines do not get stems extending to the middle staff line.

`non-break-align-symbols` (list)
: A list of symbols that determine which NON-break-aligned interfaces to align this to.

`non-default` (boolean)
: Set for manually specified clefs.

`non-musical` (boolean)
: True if the grob belongs to a `NonMusicalPaperColumn`.

`nonstaff-nonstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-relatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-unrelatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`normalized-endpoints` (pair)
: Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`outside-staff-horizontal-padding` (number)
: By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)
: The padding to place between grobs when spacing according to `outside-staff-priority`. Two grobs with different `outside-staff-padding` values have the larger value of padding between them.

`outside-staff-placement-directive` (symbol)
: One of four directives telling how outside staff objects should be placed.
  - `left-to-right-greedy` – Place each successive grob from left to right.
  - `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
  - `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
  - `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

`outside-staff-priority` (number)
: If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`packed-spacing` (boolean)
: If set, the notes are spaced as tightly as possible.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`padding-pairs` (list)
: An alist mapping `(`*`name`*` . `*`name`*`)` to distances.

`page-break-penalty` (number)
: Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

`page-break-permission` (symbol)
: Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`page-turn-penalty` (number)
: Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

`page-turn-permission` (symbol)
: Instructs the page breaker on whether to put a page turn at this column. Can be `force` or `allow`.

`parenthesized` (boolean)
: Parenthesize this grob.

`positions` (pair of numbers)
: Pair of staff coordinates `(`*`left`*` . `*`right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`prefer-dotted-right` (boolean)
: For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

`protrusion` (number)
: In an arpeggio bracket, the length of the horizontal edges.

`ratio` (number)
: Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`remove-empty` (boolean)
: If set, remove group if it contains no interesting items.

`remove-first` (boolean)
: Remove the first staff of an orchestral score?

`replacement-alist` (list)
: Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.

`restore-first` (boolean)
: Print a natural before the accidental.

`rhythmic-location` (rhythmic location)
: Where (bar number, measure position) in the score.

`right-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`rotation` (list)
: Number of degrees to rotate this object, and what point to rotate around. For example, `'(45 0 0)` rotates by 45 degrees around the center of this object.

`round-up-exceptions` (list)
: A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`round-up-to-longer-rest` (boolean)
: Displays the longer multi-measure rest when the length of a measure is between two values of `usable-duration-logs`. For example, displays a breve instead of a whole in a 3/2 measure.

`rounded` (boolean)
: Decide whether lines should be drawn rounded or not.

`same-direction-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`sharp-positions` (list)
: Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`shorten-pair` (pair of numbers)
: The lengths to shorten a text-spanner on both sides, for example a pedal bracket. Positive values shorten the text-spanner, while negative values lengthen it.

`shortest-duration-space` (dimension, in staff space)
: Start with this much space for the shortest duration. This is expressed in `spacing-increment` as unit. See also Sezione “spacing-spanner-interface” in *Guida al Funzionamento Interno*.

`shortest-playing-duration` (moment)
: The duration of the shortest note playing here.

`shortest-starter-duration` (moment)
: The duration of the shortest note that starts here.

`side-axis` (number)
: If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`side-relative-direction` (direction)
: Multiply direction of `direction-source` with this to get the direction of this object.

`simple-Y` (boolean)
: Should the Y placement of a spanner disregard changes in system heights?

`size` (number)
: Size of object, relative to standard size.

`skip-quanting` (boolean)
: Should beam quanting be skipped?

`skyline-horizontal-padding` (number)
: For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`skyline-vertical-padding` (number)
: The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extents` and `extra-spacing-heights` of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`slur-padding` (number)
: Extra distance between slur and script.

`snap-radius` (number)
: The maximum distance between two objects that will cause them to snap to alignment along an axis.

`space-alist` (list)
: A table that specifies distances between prefatory items, like clef and time-signature. The format is an alist of spacing tuples: `(`*`break-align-symbol type . distance`*`)`, where *type* can be the symbols `minimum-space` or `extra-space`.

`space-to-barline` (boolean)
: If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`spacing-increment` (number)
: Add this much space for a doubled duration. Typically, the width of a note head. See also Sezione “spacing-spanner-interface” in *Guida al Funzionamento Interno*.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

  ```
  \override MultiMeasureRest
    #'spacing-pair = #'(staff-bar . staff-bar)
  ```

`spanner-id` (string)
: An identifier to distinguish concurrent spanners.

`springs-and-rods` (boolean)
: Dummy variable for triggering spacing routines.

`stacking-dir` (direction)
: Stack objects in which direction?

`staff-affinity` (direction)
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

`staff-space` (dimension, in staff space)
: Amount of space between staff lines, expressed in global `staff-space`.

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
  - `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
  - `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
  - `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list)
: The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

`stem-attachment` (pair of numbers)
: An `(x . y)` pair where the stem attaches to the notehead.

`stem-begin-position` (number)
: User override for the begin position of a stem.

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

`stemlet-length` (number)
: How long should be a stem over a rest?

`stencil` (stencil)
: The symbol to print.

`stencils` (list)
: Multiple stencils, used as intermediate value.

`strict-grace-spacing` (boolean)
: If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

`strict-note-spacing` (boolean)
: If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

`stroke-style` (string)
: Set to `"grace"` to turn stroke through flag on.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`text` (markup)
: Text markup. See Sezione "Formatting text" in *Guida alla Notazione*.

`text-direction` (direction)
: This controls the ordering of the words. The default `RIGHT` is for roman text. Arabic or Hebrew should use `LEFT`.

`thick-thickness` (number)
: Bar line thickness, measured in `line-thickness`.

`thickness` (number)
: Line thickness, generally measured in `line-thickness`.

`thin-kern` (number)
: The space after a hair-line in a bar line.

`tie-configuration` (list)
: List of `(`*`position . dir`*`)` pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

`toward-stem-shift` (number)
: Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. `0.0` means keep the default position (centered on the note head), `1.0` means centered on the stem. Interpolated values are possible.

`transparent` (boolean)
: This makes the grob invisible.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their durations. This looks better in complex polyphonic patterns.

`usable-duration-logs` (list)
: List of `duration-log`s that can be used in typesetting the grob.

`use-skylines` (boolean)
: Should skylines be used for side positioning?

`used` (boolean)
: If set, this spacing column is kept in the spacing problem.

`vertical-skylines` (pair of skylines)
: Two skylines, one above and one below this grob.

`when` (moment)
: Global time step associated with this column happen?

`whiteout` (boolean)
: If true, the grob is printed over a white background to white-out underlying material, if the grob is visible. Usually #f by default.

`width` (dimension, in staff space)
: The width of a grob measured in staff space.

`word-space` (dimension, in staff space)
: Space to insert between words in texts.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`X-positions` (pair of numbers)
: Pair of X staff coordinates of a spanner in the form `(`*`left . right`*`)`, where both *left* and *right* are in `staff-space` units of the current staff.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number)
: The vertical amount that this object is moved relative to its Y-parent.

`zigzag-length` (dimension, in staff space)
: The length of the lines of a zigzag, relative to `zigzag-width`. A value of `1` gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)
: The width of one zigzag squiggle. This number is adjusted slightly so that the glissando line can be constructed from a whole number of squiggles.

A.18 Available music functions

`absolute` [music] - *music* (music)
: Make *music* absolute. This does not actually change the music itself but rather hides it from surrounding `\relative` commands.

`acciaccatura` [music] - *music* (music)
: Create an acciaccatura from the following music expression

`accidentalStyle` [music] - *style* (symbol list)
: Set accidental style to symbol list *style* in the form '`piano-cautionary`'. If *style* has a form like '`Staff.piano-cautionary`', the settings are applied to that context. Otherwise, the context defaults to '`Staff`', except for piano styles, which use '`GrandStaff`' as a context.

`addChordShape` [void] - *key-symbol* (symbol) *tuning* (pair) *shape-definition* (string or pair)
: Add chord shape *shape-definition* to the *chord-shape-table* hash with the key `(cons` *`key-symbol tuning`*`)`.

`addInstrumentDefinition` [void] - *name* (string) *lst* (list)
: Create instrument *name* with properties *list*.

`addQuote` [void] - *name* (string) *music* (music)
: Define *music* as a quotable music expression named *name*

`afterGrace` [music] - *main* (music) *grace* (music)
Create *grace* note(s) after a *main* music expression.

`allowPageTurn` [music]
Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

`allowVoltaHook` [void] - *bar* (string)
(undocumented; fixme)

`alterBroken` [music] - *property* (symbol list or symbol) *arg* (list) *item* (symbol list or music)
Override *property* for pieces of broken spanner *item* with values *arg*. *item* may either be music in the form of a starting spanner event, or a symbol list in the form '`Context.Grob`' or just '`Grob`'. Iff *item* is in the form of a spanner event, *property* may also have the form '`Grob.property`' for specifying a directed tweak.

`appendToTag` [music] - *tag* (symbol) *more* (music) *music* (music)
Append *more* to the `elements` of all music expressions in *music* that are tagged with *tag*.

`applyContext` [music] - *proc* (procedure)
Modify context properties with Scheme procedure *proc*.

`applyMusic` [music] - *func* (procedure) *music* (music)
Apply procedure *func* to *music*.

`applyOutput` [music] - *ctx* (symbol) *proc* (procedure)
Apply function `proc` to every layout object in context `ctx`

`appoggiatura` [music] - *music* (music)
Create an appoggiatura from *music*

`assertBeamQuant` [music] - *l* (pair) *r* (pair)
Testing function: check whether the beam quants *l* and *r* are correct

`assertBeamSlope` [music] - *comp* (procedure)
Testing function: check whether the slope of the beam is the same as `comp`

`autochange` [music] - *music* (music)
Make voices that switch between staves automatically

`balloonGrobText` [music] - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
Attach *text* to *grob-name* at offset *offset* (use like `\once`)

`balloonText` [post event] - *offset* (pair of numbers) *text* (markup)
Attach *text* at *offset* (use like `\tweak`)

`bar` [music] - *type* (string)
Insert a bar line of type *type*

`barNumberCheck` [music] - *n* (integer)
Print a warning if the current bar number is not *n*.

`bendAfter` [post event] - *delta* (real number)
Create a fall or doit of pitch interval *delta*.

`bookOutputName` [void] - *newfilename* (string)
Direct output for the current book block to *newfilename*.

`bookOutputSuffix` [void] - *newsuffix* (string)
Set the output filename suffix for the current book block to *newsuffix*.

`breathe` [music]
: Insert a breath mark.

`chordRepeats` [music] - *event-types* [list] *music* (music)
: Walk through *music* putting the notes of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(string-number-event)`.

`clef` [music] - *type* (string)
: Set the current clef to *type*.

`compoundMeter` [music] - *args* (pair)
: Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the enumerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of (3+1)/8 + 2/4 would be created as `\compoundMeter #'((3 1 8) (2 4))`, and a time signature of (3+2)/8 as `\compoundMeter #'((3 2 8))` or shorter `\compoundMeter #'(3 2 8)`.

`crossStaff` [music] - *notes* (music)
: Create cross-staff stems

`cueClef` [music] - *type* (string)
: Set the current cue clef to *type*.

`cueClefUnset` [music]
: Unset the current cue clef.

`cueDuring` [music] - *what* (string) *dir* (direction) *main-music* (music)
: Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`cueDuringWithClef` [music] - *what* (string) *dir* (direction) *clef* (string) *main-music* (music)
: Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`deadNote` [music] - *note* (music)
: Print *note* with a cross-shaped note head.

`defaultNoteHeads` [music]
: Revert to the default note head style.

`defineBarLine` [void] - *bar* (string) *glyph-list* (list)
: Define bar line settings for bar line *bar*. The list *glyph-list* must have three entries which define the appearance at the end of line, at the beginning of the next line, and the span bar, respectively.

`displayLilyMusic` [music] - *music* (music)
: Display the LilyPond input representation of *music* to the console.

`displayMusic` [music] - *music* (music)
: Display the internal representation of *music* to the console.

`displayScheme` (any type) - *expr* (any type)
: Display the internal representation of *expr* to the console.

`endSpanners` [music] - *music* (music)
: Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.

`eventChords` [music] - *music* (music)
: Compatibility function wrapping `EventChord` around isolated rhythmic events occuring since version 2.15.28, after expanding repeat chords '`q`'.

`featherDurations` [music] - *factor* (moment) *argument* (music)
: Adjust durations of music in *argument* by rational *factor*.

`finger` [post event] - *finger* (number or markup)
: Apply *finger* as a fingering indication.

`footnote` [music] - *mark* [markup] *offset* (pair of numbers) *footnote* (markup) *item* (symbol list or music)
: Make the markup *footnote* a footnote on *item*. The footnote is marked with a markup *mark* moved by *offset* with respect to the marked music.

  If *mark* is not given or specified as `\default`, it is replaced by an automatically generated sequence number. If *item* is a symbol list of form '`Grob`' or '`Context.Grob`', then grobs of that type will be marked at the current time step in the given context (default `Bottom`).

  If *item* is music, the music will get a footnote attached to a grob immediately attached to the event, like `\tweak` does. For attaching a footnote to an *indirectly* caused grob, write `\single\footnote`, use *item* to specify the grob, and follow it with the music to annotate.

  Like with `\tweak`, if you use a footnote on a following post-event, the `\footnote` command itself needs to be attached to the preceding note or rest as a post-event with `-`.

`grace` [music] - *music* (music)
: Insert *music* as grace notes.

`grobdescriptions` (any type) - *descriptions* (list)
: Create a context modification from *descriptions*, a list in the format of `all-grob-descriptions`.

`harmonicByFret` [music] - *fret* (number) *music* (music)
: Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at *fret*.

`harmonicByRatio` [music] - *ratio* (number) *music* (music)
: Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at the point given through *ratio*.

`harmonicNote` [music] - *note* (music)
: Print *note* with a diamond-shaped note head.

`harmonicsOn` [music]
: Set the default note head style to a diamond-shaped style.

`hide` [music] - *item* (symbol list or music)
: Set *item*'s '`transparent`' property to `#t`, making it invisible while still retaining its dimensions.

  If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`inStaffSegno` [music]
: Put the segno variant 'varsegno' at this position into the staff, compatible with the repeat command.

`instrumentSwitch` [music] - *name* (string)
: Switch instrument to *name*, which must be predefined with `\addInstrumentDefinition`.

`inversion` [music] - *around* (pitch) *to* (pitch) *music* (music)
: Invert *music* about *around* and transpose from *around* to *to*.

`keepWithTag` [music] - *tag* (symbol list or symbol) *music* (music)
: Include only elements of *music* that are either untagged or tagged with one of the tags in *tag*. *tag* may be either a single symbol or a list of symbols.

`key` [music] - *tonic* [pitch] *pitch-alist* [list]
: Set key to *tonic* and scale *pitch-alist*. If both are null, just generate `KeyChangeEvent`.

`killCues` [music] - *music* (music)
: Remove cue notes from *music*.

`label` [music] - *label* (symbol)
: Create *label* as a bookmarking label.

`language` [void] - *language* (string)
: Set note names for language *language*.

`languageRestore` [void]
: Restore a previously-saved pitchnames alist.

`languageSaveAndChange` [void] - *language* (string)
: Store the previous pitchnames alist, and set a new one.

`makeClusters` [music] - *arg* (music)
: Display chords in *arg* as clusters.

`makeDefaultStringTuning` [void] - *symbol* (symbol) *pitches* (list)
: This defines a string tuning *symbol* via a list of *pitches*. The *symbol* also gets registered in `defaultStringTunings` for documentation purposes.

`mark` [music] - *label* [any type]
: Make the music for the \mark command.

`modalInversion` [music] - *around* (pitch) *to* (pitch) *scale* (music) *music* (music)
: Invert *music* about *around* using *scale* and transpose from *around* to *to*.

`modalTranspose` [music] - *from* (pitch) *to* (pitch) *scale* (music) *music* (music)
: Transpose *music* from pitch *from* to pitch *to* using *scale*.

`musicMap` [music] - *proc* (procedure) *mus* (music)
: Apply *proc* to *mus* and all of the music it contains.

`noPageBreak` [music]
: Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`noPageTurn` [music]
: Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.

`octaveCheck` [music] - *pitch* (pitch)
: Octave check.

`offset` [music] - *property* (symbol list or symbol) *offsets* (any type) *item* (symbol list or music)
: Offset the default value of *property* of *item* by *offsets*. If *item* is a string, the result is `\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`omit` [music] - *item* (symbol list or music)
: Set *item*'s '`stencil`' property to `#f`, effectively omitting it without taking up space.

  If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`once` [music] - *music* (music)
: Set `once` to `#t` on all layout instruction events in *music*. This will complain about music with an actual duration. As a special exception, if *music* contains '`tweaks`' it will be silently ignored in order to allow for `\once \tweak` to work as both one-time override and proper tweak.

`ottava` [music] - *octave* (integer)
: Set the octavation.

`overrideProperty` [music] - *grob-property-path* (symbol list) *value* (any type)
: Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well.

`overrideTimeSignatureSettings` [music] - *time-signature* (fraction, as pair) *base-moment* (fraction, as pair) *beat-structure* (list) *beam-exceptions* (list)
: Override `timeSignatureSettings` for time signatures of *time-signature* to have settings of *base-moment*, *beat-structure*, and *beam-exceptions*.

`pageBreak` [music]
: Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`pageTurn` [music]
: Force a page turn between two scores or top-level markups.

`palmMute` [music] - *note* (music)
: Print *note* with a triangle-shaped note head.

`palmMuteOn` [music]
: Set the default note head style to a triangle-shaped style.

`parallelMusic` [void] - *voice-ids* (list) *music* (music)
: Define parallel music sequences, separated by '`|`' (bar check signs), and assign them to the identifiers provided in *voice-ids*.

  voice-ids: a list of music identifiers (symbols containing only letters)

  music: a music sequence, containing BarChecks as limiting expressions.

  Example:

```
\parallelMusic #'(A B C) {
  c c | d d | e e |
  d d | e e | f f |
}
<==>
A = { c c | d d | }
B = { d d | e e | }
C = { e e | f f | }
```

`parenthesize` [music] - *arg* (music)
: Tag *arg* to be parenthesized.

`partcombine` [music] - *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff.

`partcombineDown` [music] - *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed downward.

`partcombineForce` [music] - *type* (symbol-or-boolean) *once* (boolean)
: Override the part-combiner.

`partcombineUp` [music] - *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed upward.

`partial` [music] - *dur* (duration)
: Make a partial measure.

`phrasingSlurDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for phrasing slurs.

`pitchedTrill` [music] - *main-note* (music) *secondary-note* (music)
: Print a trill with *main-note* as the main note of the trill and print *secondary-note* as a stemless note head in parentheses.

`pointAndClickOff` [void]
: Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

`pointAndClickOn` [void]
: Enable generation of code in final-format (e.g. pdf) files to reference the originating lilypond source statement; this is helpful when developing a score but generates bigger final-format files.

`pointAndClickTypes` [void] - *types* (symbol list or symbol)
: Set a type or list of types (such as `#'note-event`) for which point-and-click info is generated.

`pushToTag` [music] - *tag* (symbol) *more* (music) *music* (music)
: Add *more* to the front of `elements` of all music expressions in *music* that are tagged with *tag*.

`quoteDuring` [music] - *what* (string) *main-music* (music)
: Indicate a section of music to be quoted. *what* indicates the name of the quoted voice, as specified in an `\addQuote` command. *main-music* is used to indicate the length of music to be quoted; usually contains spacers or multi-measure rests.

`relative` [music] - *pitch* [pitch] *music* (music)
: Make *music* relative to *pitch*. If *pitch* is omitted, the first note in *music* is given in absolute pitch.

`removeWithTag` [music] - *tag* (symbol list or symbol) *music* (music)
: Remove elements of *music* that are tagged with one of the tags in *tag*. *tag* may be either a single symbol or a list of symbols.

`resetRelativeOctave` [music] - *pitch* (pitch)
: Set the octave inside a \relative section.

`retrograde` [music] - *music* (music)
: Return *music* in reverse order.

`revertTimeSignatureSettings` [music] - *time-signature* (pair)
: Revert `timeSignatureSettings` for time signatures of *time-signature*.

`rightHandFinger` [post event] - *finger* (number or markup)
: Apply *finger* as a fingering indication.

`scaleDurations` [music] - *fraction* (fraction, as pair) *music* (music)
: Multiply the duration of events in *music* by *fraction*.

`settingsFrom` (any type) - *ctx* [symbol] *music* (music)
: Take the layout instruction events from *music*, optionally restricted to those applying to context type *ctx*, and return a context modification duplicating their effect.

`shape` [music] - *offsets* (list) *item* (symbol list or music)
: Offset control-points of *item* by *offsets*. The argument is a list of number pairs or list of such lists. Each element of a pair represents an offset to one of the coordinates of a control-point. If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`shiftDurations` [music] - *dur* (integer) *dots* (integer) *arg* (music)
: Change the duration of *arg* by adding *dur* to the `durlog` of *arg* and *dots* to the `dots` of *arg*.

`single` [music] - *overrides* (music) *music* (music)
: Convert *overrides* to tweaks and apply them to *music*. This does not convert `\revert`, `\set` or `\unset`.

`skip` [music] - *dur* (duration)
: Skip forward by *dur*.

`slashedGrace` [music] - *music* (music)
: Create slashed graces (slashes through stems, but no slur) from the following music expression

`slurDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for slurs.

`spacingTweaks` [music] - *parameters* (list)
: Set the system stretch, by reading the 'system-stretch property of the 'parameters' assoc list.

`storePredefinedDiagram` [void] - *fretboard-table* (hash table) *chord* (music) *tuning* (pair) *diagram-definition* (string or pair)
: Add predefined fret diagram defined by *diagram-definition* for the chord pitches *chord* and the stringTuning *tuning*.

`stringTuning` (any type) - *chord* (music)
: Convert *chord* to a string tuning. *chord* must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.

`styledNoteHeads` [music] - *style* (symbol) *heads* (symbol list or symbol) *music* (music)
: Set *heads* in *music* to *style*.

`tabChordRepeats` [music] - *event-types* [list] *music* (music)
: Walk through *music* putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(articulation-event)`.

`tabChordRepetition` [void]
: Include the string and fingering information in a chord repetition. This function is deprecated; try using `\tabChordRepeats` instead.

`tag` [music] - *tag* (symbol list or symbol) *music* (music)
Tag the following *music* with *tag* and return the result, by adding the single symbol or symbol list *tag* to the `tags` property of *music*.

`temporary` [music] - *music* (music)
Make any `\override` in *music* replace an existing grob property value only temporarily, restoring the old value when a corresponding `\revert` is executed. This is achieved by clearing the '`pop-first`' property normally set on `\override`s.

An `\override`/`\revert` sequence created by using `\temporary` and `\undo` on the same music containing overrides will cancel out perfectly or cause a warning.

Non-property-related music is ignored, warnings are generated for any property-changing music that isn't an `\override`.

`tieDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for ties.

`time` [music] - *beat-structure* [number list] *fraction* (fraction, as pair)
Set *fraction* as time signature, with optional number list *beat-structure* before it.

`times` [music] - *fraction* (fraction, as pair) *music* (music)
Scale *music* in time by *fraction*.

`tocItem` [music] - *text* (markup)
Add a line to the table of content, using the `tocItemMarkup` paper variable markup

`transpose` [music] - *from* (pitch) *to* (pitch) *music* (music)
Transpose *music* from pitch *from* to pitch *to*.

`transposedCueDuring` [music] - *what* (string) *dir* (direction) *pitch* (pitch) *main-music* (music)
Insert notes from the part *what* into a voice called `cue`, using the transposition defined by *pitch*. This happens simultaneously with *main-music*, which is usually a rest. The argument *dir* determines whether the cue notes should be notated as a first or second voice.

`transposition` [music] - *pitch* (pitch)
Set instrument transposition

`tuplet` [music] - *ratio* (fraction, as pair) *tuplet-span* [duration] *music* (music)
Scale the given *music* to tuplets. *ratio* is a fraction that specifies how many notes are played in place of the nominal value: it will be '`3/2`' for triplets, namely three notes being played in place of two. If the optional duration *tuplet-span* is specified, it is used instead of `tupletSpannerDuration` for grouping the tuplets. For example,

```
\tuplet 3/2 4 { c8 c c c c c }
```

will result in two groups of three tuplets, each group lasting for a quarter note.

`tupletSpan` [music] - *tuplet-span* [duration]
Set `tupletSpannerDuration`, the length into which `\tuplet` without an explicit '`tuplet-span`' argument of its own will group its tuplets, to the duration *tuplet-span*. To revert to the default of not subdividing the contents of a `\tuplet` command without explicit '`tuplet-span`', use

```
\tupletSpan \default
```

`tweak` [music] - *prop* (symbol list or symbol) *value* (any type) *item* (symbol list or music)
Add a tweak to the following *item*, usually music. Layout objects created by *item* get their property *prop* set to *value*. If *prop* has the form '`Grob.property`', like with

`\tweak Accidental.color #red cis'`

an indirectly created grob ('`Accidental`' is caused by '`NoteHead`') can be tweaked; otherwise only directly created grobs are affected.

As a special case, *item* may be a symbol list specifying a grob path, in which case `\override` is called on it instead of creating tweaked music. This is mainly useful when using `\tweak` as as a component for building other functions.

If this use case would call for `\once \override` rather than a plain `\override`, writing `\once \tweak ...` can be convenient.

prop can contain additional elements in which case a nested property (inside of an alist) is tweaked.

`undo` [music] - *music* (music)
: Convert `\override` and `\set` in *music* to `\revert` and `\unset`, respectively. Any reverts and unsets already in *music* cause a warning. Non-property-related music is ignored.

`unfoldRepeats` [music] - *music* (music)
: Force any `\repeat volta`, `\repeat tremolo` or `\repeat percent` commands in *music* to be interpreted as `\repeat unfold`.

`void` [void] - *arg* (any type)
: Accept a scheme argument, return a void expression. Use this if you want to have a scheme expression evaluated because of its side-effects, but its value ignored.

`withMusicProperty` [music] - *sym* (symbol) *val* (any type) *music* (music)
: Set *sym* to *val* in *music*.

`xNote` [music] - *note* (music)
: Print *note* with a cross-shaped note head.

`xNotesOn` [music]
: Set the default note head style to a cross-shaped style.

A.19 Context modification identifiers

The following commands are defined for use as context modifications within a `\layout` or `\with` block.

`RemoveEmptyStaves`
: Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`.
  - Sets grob property `remove-empty` in Sezione ''VerticalAxisGroup'' in *Guida al Funzionamento Interno* to `#t`.

A.20 Predefined type predicates

R5RS primary predicates

Type predicate	Description
`boolean?`	boolean
`char?`	character
`number?`	number
`pair?`	pair
`port?`	port
`procedure?`	procedure

`string?`	string
`symbol?`	symbol
`vector?`	vector

R5RS secondary predicates

Type predicate	**Description**
`char-alphabetic?`	alphabetic character
`char-lower-case?`	lower-case character
`char-numeric?`	numeric character
`char-upper-case?`	upper-case character
`char-whitespace?`	whitespace character
`complex?`	complex number
`eof-object?`	end-of-file object
`even?`	even number
`exact?`	exact number
`inexact?`	inexact number
`input-port?`	input port
`integer?`	integer
`list?`	list *(use* `cheap-list?` *for faster processing)*
`negative?`	negative number
`null?`	null
`odd?`	odd number
`output-port?`	output port
`positive?`	positive number
`rational?`	rational number
`real?`	real number
`zero?`	zero

Guile predicates

Type predicate	**Description**
`hash-table?`	hash table

LilyPond scheme predicates

Type predicate	**Description**
`boolean-or-symbol?`	boolean or symbol
`cheap-list?`	list *(use this instead of* `list?` *for faster processing)*
`color?`	color
`fraction?`	fraction, as pair
`grob-list?`	list of grobs
`index?`	non-negative integer
`markup?`	markup
`markup-command-list?`	markup command list
`markup-list?`	markup list
`moment-pair?`	pair of moment objects
`number-list?`	number list
`number-or-grob?`	number or grob
`number-or-markup?`	number or markup
`number-or-pair?`	number or pair

number-or-string?	number or string
number-pair?	pair of numbers
number-pair-list?	list of number pairs
rhythmic-location?	rhythmic location
scheme?	any type
string-or-music?	string or music
string-or-pair?	string or pair
string-or-symbol?	string or symbol
symbol-list?	symbol list
symbol-list-or-music?	symbol list or music
symbol-list-or-symbol?	symbol list or symbol
void?	void

LilyPond exported predicates

Type predicate	**Description**
ly:book?	book
ly:box?	box
ly:context?	context
ly:context-def?	context definition
ly:context-mod?	context modification
ly:dimension?	dimension, in staff space
ly:dir?	direction
ly:dispatcher?	dispatcher
ly:duration?	duration
ly:event?	post event
ly:font-metric?	font metric
ly:grob?	graphical (layout) object
ly:grob-array?	array of grobs
ly:input-location?	input location
ly:item?	item
ly:iterator?	iterator
ly:lily-lexer?	lily-lexer
ly:lily-parser?	lily-parser
ly:listener?	listener
ly:moment?	moment
ly:music?	music
ly:music-function?	music function
ly:music-list?	list of music objects
ly:music-output?	music output
ly:otf-font?	OpenType font
ly:output-def?	output definition
ly:page-marker?	page marker
ly:pango-font?	pango font
ly:paper-book?	paper book
ly:paper-system?	paper-system Prob
ly:pitch?	pitch
ly:prob?	property object
ly:score?	score
ly:simple-closure?	simple closure
ly:skyline?	skyline

`ly:skyline-pair?`	pair of skylines
`ly:source-file?`	source file
`ly:spanner?`	spanner
`ly:spring?`	spring
`ly:stencil?`	stencil
`ly:stream-event?`	stream event
`ly:translator?`	translator
`ly:translator-group?`	translator group
`ly:unpure-pure-container?`	unpure/pure container

A.21 Scheme functions

`ly:add-context-mod` *contextmods modification* [Funzione]
Adds the given context *modification* to the list *contextmods* of context modifications.

`ly:add-file-name-alist` *alist* [Funzione]
Add mappings for error messages from *alist*.

`ly:add-interface` *iface desc props* [Funzione]
Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

`ly:add-listener` *list disp cl* [Funzione]
Add the listener *list* to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it is forwarded to *list*.

`ly:add-option` *sym val description* [Funzione]
Add a program option *sym*. *val* is the default value and *description* is a string description.

`ly:all-grob-interfaces` [Funzione]
Return the hash table with all grob interface descriptions.

`ly:all-options` [Funzione]
Get all option settings in an alist.

`ly:all-stencil-expressions` [Funzione]
Return all symbols recognized as stencil expressions.

`ly:assoc-get` *key alist default-value strict-checking* [Funzione]
Return value if *key* in *alist*, else *default-value* (or `#f` if not specified). If *strict-checking* is set to `#t` and *key* is not in *alist*, a programming_error is output.

`ly:axis-group-interface::add-element` *grob grob-element* [Funzione]
Set *grob* the parent of *grob-element* on all axes of *grob*.

`ly:basic-progress` *str rest* [Funzione]
A Scheme callable function to issue a basic progress message *str*. The message is formatted with `format` and *rest*.

`ly:beam-score-count` [Funzione]
count number of beam scores.

`ly:book?` *x* [Funzione]
Is *x* a `Book` object?

`ly:book-add-bookpart!` *book-smob book-part* [Funzione]
Add *book-part* to *book-smob* book part list.

`ly:book-add-score!` *book-smob score* [Funzione]
Add *score* to *book-smob* score list.

`ly:book-book-parts` *book* [Funzione]
Return book parts in *book*.

`ly:book-header` *book* [Funzione]
Return header in *book*.

`ly:book-paper` *book* [Funzione]
Return paper in *book*.

`ly:book-process` *book-smob default-paper default-layout output* [Funzione]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-process-to-systems` *book-smob default-paper default-layout output* [Funzione]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-scores` *book* [Funzione]
Return scores in *book*.

`ly:book-set-header!` *book module* [Funzione]
Set the book header.

`ly:box?` *x* [Funzione]
Is *x* a `Box` object?

`ly:bp` *num* [Funzione]
num bigpoints (1/72th inch).

`ly:bracket` *a iv t p* [Funzione]
Make a bracket in direction *a*. The extent of the bracket is given by *iv*. The wings protrude by an amount of *p*, which may be negative. The thickness is given by *t*.

`ly:broadcast` *disp ev* [Funzione]
Send the stream event *ev* to the dispatcher *disp*.

`ly:camel-case->lisp-identifier` *name-sym* [Funzione]
Convert `FooBar_Bla` to `foo-bar-bla` style symbol.

`ly:chain-assoc-get` *key achain default-value strict-checking* [Funzione]
Return value for *key* from a list of alists *achain*. If no entry is found, return *default-value* or `#f` if *default-value* is not specified. With *strict-checking* set to `#t`, a programming_error is output in such cases.

`ly:check-expected-warnings` [Funzione]
Check whether all expected warnings have really been triggered.

`ly:cm` *num* [Funzione]
num cm.

`ly:command-line-code` [Funzione]
The Scheme code specified on command-line with '`-e`'.

`ly:command-line-options` [Funzione]
The Scheme options specified on command-line with '`-d`'.

`ly:connect-dispatchers` *to from* [Funzione]
Make the dispatcher *to* listen to events from *from*.

`ly:context?` *x* [Funzione]
Is *x* a `Context` object?

`ly:context-current-moment` *context* [Funzione]
Return the current moment of *context*.

`ly:context-def?` *x* [Funzione]
Is *x* a `Context_def` object?

`ly:context-def-lookup` *def sym val* [Funzione]
Return the value of *sym* in context definition *def* (e.g., `\Voice`). If no value is found, return *val* or `'()` if *val* is undefined. *sym* can be any of '`default-child`', '`consists`', '`description`', '`aliases`', '`accepts`', '`property-ops`', '`context-name`', '`group-type`'.

`ly:context-def-modify` *def mod* [Funzione]
Return the result of applying the context-mod *mod* to the context definition *def*. Does not change *def*.

`ly:context-event-source` *context* [Funzione]
Return `event-source` of context *context*.

`ly:context-events-below` *context* [Funzione]
Return a `stream-distributor` that distributes all events from *context* and all its subcontexts.

`ly:context-find` *context name* [Funzione]
Find a parent of *context* that has name or alias *name*. Return `#f` if not found.

`ly:context-grob-definition` *context name* [Funzione]
Return the definition of *name* (a symbol) within *context* as an alist.

`ly:context-id` *context* [Funzione]
Return the ID string of *context*, i.e., for `\context Voice = "one" ...` return the string `one`.

`ly:context-mod?` *x* [Funzione]
Is *x* a `Context_mod` object?

`ly:context-mod-apply!` *context mod* [Funzione]
Apply the context modification *mod* to *context*.

`ly:context-name` *context* [Funzione]
Return the name of *context*, i.e., for `\context Voice = "one" ...` return the symbol `Voice`.

`ly:context-now` *context* [Funzione]
Return `now-moment` of context *context*.

`ly:context-parent` *context* [Funzione]
Return the parent of *context*, `#f` if none.

`ly:context-property` *context sym def* [Funzione]
Return the value for property *sym* in *context*. If *def* is given, and property value is `'()`, return *def*.

`ly:context-property-where-defined` *context name* [Funzione]
Return the context above *context* where *name* is defined.

`ly:context-pushpop-property` *context grob eltprop val* [Funzione]
Do a single `\override` or `\revert` operation in *context*. The grob definition *grob* is extended with *eltprop* (if *val* is specified) or reverted (if unspecified).

`ly:context-set-property!` *context name val* [Funzione]
Set value of property *name* in context *context* to *val*.

`ly:context-unset-property` *context name* [Funzione]
Unset value of property *name* in context *context*.

`ly:debug` *str rest* [Funzione]
A Scheme callable function to issue a debug message *str*. The message is formatted with `format` and *rest*.

`ly:default-scale` [Funzione]
Get the global default scale.

`ly:dimension?` *d* [Funzione]
Return *d* as a number. Used to distinguish length variables from normal numbers.

`ly:dir?` *s* [Funzione]
Is *s* a direction? Valid directions are `-1`, `0`, or `1`, where `-1` represents left or down, `1` represents right or up, and `0` represents a neutral direction.

`ly:dispatcher?` *x* [Funzione]
Is *x* a `Dispatcher` object?

`ly:duration?` *x* [Funzione]
Is *x* a `Duration` object?

`ly:duration<?` *p1 p2* [Funzione]
Is *p1* shorter than *p2*?

`ly:duration->string` *dur* [Funzione]
Convert *dur* to a string.

`ly:duration-dot-count` *dur* [Funzione]
Extract the dot count from *dur*.

`ly:duration-factor` *dur* [Funzione]
Extract the compression factor from *dur*. Return it as a pair.

`ly:duration-length` *dur* [Funzione]
The length of the duration as a `moment`.

`ly:duration-log` *dur* [Funzione]
Extract the duration log from *dur*.

`ly:duration-scale` *dur* [Funzione]
Extract the compression factor from *dur*. Return it as a rational.

`ly:effective-prefix` [Funzione]
Return effective prefix.

`ly:encode-string-for-pdf` *str* [Funzione]
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that's not possible to full UTF-16BE with Byte-Order-Mark (BOM).

`ly:engraver-announce-end-grob` *engraver grob cause* [Funzione]
Announce the end of a grob (i.e., the end of a spanner) originating from given *engraver* instance, with *grob* being a grob. *cause* should either be another grob or a music event.

`ly:engraver-make-grob` *engraver grob-name cause* [Funzione]
Create a grob originating from given *engraver* instance, with given *grob-name*, a symbol. *cause* should either be another grob or a music event.

`ly:error` *str rest* [Funzione]
A Scheme callable function to issue the error *str*. The error is formatted with `format` and *rest*.

`ly:eval-simple-closure` *delayed closure scm-start scm-end* [Funzione]
Evaluate a simple *closure* with the given *delayed* argument. If *scm-start* and *scm-end* are defined, evaluate it purely with those start and end points.

`ly:event?` *obj* [Funzione]
Is *obj* a proper (non-rhythmic) event object?

`ly:event-deep-copy` *m* [Funzione]
Copy *m* and all sub expressions of *m*.

`ly:event-property` *sev sym val* [Funzione]
Get the property *sym* of stream event *sev*. If *sym* is undefined, return *val* or `'()` if *val* is not specified.

`ly:event-set-property!` *ev sym val* [Funzione]
Set property *sym* in event *ev* to *val*.

`ly:expand-environment` *str* [Funzione]
Expand `$VAR` and `${VAR}` in *str*.

`ly:expect-warning` *str rest* [Funzione]
A Scheme callable function to register a warning to be expected and subsequently suppressed. If the warning is not encountered, a warning about the missing warning will be shown. The message should be translated with `(_ ...)` and changing parameters given after the format string.

`ly:find-file` *name* [Funzione]
Return the absolute file name of *name*, or `#f` if not found.

`ly:font-config-add-directory` *dir* [Funzione]
Add directory *dir* to FontConfig.

`ly:font-config-add-font` *font* [Funzione]
Add font *font* to FontConfig.

`ly:font-config-display-fonts` [Funzione]
Dump a list of all fonts visible to FontConfig.

`ly:font-config-get-font-file` *name* [Funzione]
Get the file for font *name*.

`ly:font-design-size` *font* [Funzione]
Given the font metric *font*, return the design size, relative to the current output-scale.

`ly:font-file-name` *font* [Funzione]
Given the font metric *font*, return the corresponding file name.

`ly:font-get-glyph` *font name* [Funzione]
Return a stencil from *font* for the glyph named *name*. If the glyph is not available, return an empty stencil.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-charcode` *font name* [Funzione]
Return the character code for glyph *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-index` *font name* [Funzione]
Return the index for *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-index-to-charcode` *font index* [Funzione]
Return the character code for *index* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-magnification` *font* [Funzione]
Given the font metric *font*, return the magnification, relative to the current output-scale.

`ly:font-metric?` *x* [Funzione]
Is *x* a `Font_metric` object?

`ly:font-name` *font* [Funzione]
Given the font metric *font*, return the corresponding name.

`ly:font-sub-fonts` *font* [Funzione]
Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

`ly:format` *str rest* [Funzione]
LilyPond specific format, supporting `~a` and `~[0-9]f`. Basic support for `~s` is also provided.

`ly:format-output` *context* [Funzione]
Given a global context in its final state, process it and return the `Music_output` object in its final state.

`ly:get-all-function-documentation` [Funzione]
Get a hash table with all LilyPond Scheme extension functions.

`ly:get-all-translators` [Funzione]
Return a list of all translator objects that may be instantiated.

`ly:get-context-mods` *contextmod* [Funzione]
Returns the list of context modifications stored in *contextmod*.

`ly:get-option` *var* [Funzione]
Get a global option setting.

`ly:get-spacing-spec` *from-scm to-scm* [Funzione]
Return the spacing spec going between the two given grobs, *from_scm* and *to_scm*.

`ly:get-undead` *undead* [Funzione]
Get back object from *undead*.

`ly:gettext` *original* [Funzione]
A Scheme wrapper function for `gettext`.

`ly:grob?` *x* [Funzione]
Is *x* a `Grob` object?

`ly:grob-alist-chain` *grob global* [Funzione]
Get an alist chain for grob *grob*, with *global* as the global default. If unspecified, `font-defaults` from the layout block is taken.

`ly:grob-array?` *x* [Funzione]
Is *x* a `Grob_array` object?

`ly:grob-array->list` *grob-arr* [Funzione]
Return the elements of *grob-arr* as a Scheme list.

`ly:grob-array-length` *grob-arr* [Funzione]
Return the length of *grob-arr*.

`ly:grob-array-ref` *grob-arr index* [Funzione]
Retrieve the *index*th element of *grob-arr*.

`ly:grob-basic-properties` *grob* [Funzione]
Get the immutable properties of *grob*.

`ly:grob-chain-callback` *grob proc sym* [Funzione]
Find the callback that is stored as property *sym* of grob *grob* and chain *proc* to the head of this, meaning that it is called using *grob* and the previous callback's result.

`ly:grob-common-refpoint` *grob other axis* [Funzione]
Find the common refpoint of *grob* and *other* for *axis*.

`ly:grob-common-refpoint-of-array` *grob others axis* [Funzione]
Find the common refpoint of *grob* and *others* (a grob-array) for *axis*.

`ly:grob-default-font` *grob* [Funzione]
Return the default font for grob *grob*.

`ly:grob-extent` *grob refp axis* [Funzione]
Get the extent in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-get-vertical-axis-group-index` *grob* [Funzione]
Get the index of the vertical axis group the grob *grob* belongs to; return `-1` if none is found.

`ly:grob-interfaces` *grob* [Funzione]
Return the interfaces list of grob *grob*.

`ly:grob-layout` *grob* [Funzione]
Get `\layout` definition from grob *grob*.

`ly:grob-object` *grob sym* [Funzione]
Return the value of a pointer in grob *grob* of property *sym*. It returns `'()` (end-of-list) if *sym* is undefined in *grob*.

`ly:grob-original` *grob* [Funzione]
Return the unbroken original grob of *grob*.

`ly:grob-parent` *grob axis* [Funzione]
Get the parent of *grob*. *axis* is 0 for the X-axis, 1 for the Y-axis.

`ly:grob-pq<?` *a b* [Funzione]
Compare two grob priority queue entries. This is an internal function.

`ly:grob-properties` *grob* [Funzione]
Get the mutable properties of *grob*.

`ly:grob-property` *grob sym val* [Funzione]
Return the value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-property-data` *grob sym* [Funzione]
Return the value for property *sym* of *grob*, but do not process callbacks.

`ly:grob-pure-height` *grob refp beg end val* [Funzione]
Return the pure height of *grob* given refpoint *refp*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-pure-property` *grob sym beg end val* [Funzione]
Return the pure value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-relative-coordinate` *grob refp axis* [Funzione]
Get the coordinate in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-robust-relative-extent` *grob refp axis* [Funzione]
Get the extent in *axis* direction of *grob* relative to the grob *refp*, or `(0,0)` if empty.

`ly:grob-script-priority-less` *a b* [Funzione]
Compare two grobs by script priority. For internal use.

`ly:grob-set-nested-property!` *grob symlist val* [Funzione]
Set nested property *symlist* in grob *grob* to value *val*.

`ly:grob-set-object!` *grob sym val* [Funzione]
Set *sym* in grob *grob* to value *val*.

`ly:grob-set-parent!` *grob axis parent-grob* [Funzione]
Set *parent-grob* the parent of grob *grob* in axis *axis*.

`ly:grob-set-property!` *grob sym val* [Funzione]
Set *sym* in grob *grob* to value *val*.

`ly:grob-staff-position` *sg* [Funzione]
Return the Y-position of *sg* relative to the staff.

`ly:grob-suicide!` *grob* [Funzione]
Kill *grob*.

`ly:grob-system` *grob* [Funzione]
Return the system grob of *grob*.

`ly:grob-translate-axis!` *grob d a* [Funzione]
Translate *grob* on axis *a* over distance *d*.

`ly:grob-vertical<?` *a b* [Funzione]
Does *a* lie above *b* on the page?

`ly:gulp-file` *name size* [Funzione]
Read *size* characters from the file *name*, and return its contents in a string. If *size* is undefined, the entire file is read. The file is looked up using the search path.

`ly:hash-table-keys` *tab* [Funzione]
Return a list of keys in *tab*.

`ly:inch` *num* [Funzione]
num inches.

`ly:input-both-locations` *sip* [Funzione]
Return input location in *sip* as `(file-name first-line first-column last-line last-column)`.

`ly:input-file-line-char-column` *sip* [Funzione]
Return input location in *sip* as `(file-name line char column)`.

`ly:input-location?` *x* [Funzione]
Is *x* an `input-location`?

`ly:input-message` *sip msg rest* [Funzione]
Print *msg* as a GNU compliant error message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:input-warning` *sip msg rest* [Funzione]
Print *msg* as a GNU compliant warning message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:interpret-music-expression` *mus ctx* [Funzione]
Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.

`ly:interpret-stencil-expression` *expr func arg1 offset* [Funzione]
Parse *expr*, feed bits to *func* with first arg *arg1* having offset *offset*.

`ly:intlog2` *d* [Funzione]
The 2-logarithm of $1/d$.

`ly:item?` *g* [Funzione]
Is *g* an `Item` object?

`ly:item-break-dir` *it* [Funzione]
The break status direction of item *it*. `-1` means end of line, `0` unbroken, and `1` beginning of line.

`ly:iterator?` *x* [Funzione]
Is *x* a `Music_iterator` object?

`ly:lexer-keywords` *lexer* [Funzione]
Return a list of `(KEY . CODE)` pairs, signifying the LilyPond reserved words list.

`ly:lily-lexer?` *x* [Funzione]
Is *x* a `Lily_lexer` object?

`ly:lily-parser?` *x* [Funzione]
Is *x* a `Lily_parser` object?

`ly:listened-event-class?` *disp cl* [Funzione]
Does *disp* listen to any event type in the list *cl*?

`ly:listened-event-types` *disp* [Funzione]
Return a list of all event types that *disp* listens to.

`ly:listener?` *x* [Funzione]
Is *x* a `Listener` object?

`ly:make-book` *paper header scores* [Funzione]
Make a `\book` of *paper* and *header* (which may be `#f` as well) containing `\scores`.

`ly:make-book-part` *scores* [Funzione]
Make a `\bookpart` containing `\scores`.

`ly:make-context-mod` *mod-list* [Funzione]
Creates a context modification, optionally initialized via the list of modifications *mod-list*.

`ly:make-dispatcher` [Funzione]
Return a newly created dispatcher.

`ly:make-duration` *length dotcount num den* [Funzione]
length is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.

The duration factor is optionally given by integers *num* and *den*, alternatively by a single rational number.

A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.

`ly:make-global-context` *output-def* [Funzione]
Set up a global interpretation context, using the output block *output-def*. The context is returned.

`ly:make-global-translator` *global* [Funzione]
Create a translator group and connect it to the global context *global*. The translator group is returned.

`ly:make-listener` *callback* [Funzione]
Create a listener. Any time the listener hears an object, it will call *callback* with that object.

callback should take exactly one argument.

`ly:make-moment` *m g gn gd* [Funzione]
Create the moment with rational main timing *m*, and optional grace timing *g*.

A *moment* is a point in musical time. It consists of a pair of rationals (*m*, *g*), where *m* is the timing for the main notes, and *g* the timing for grace notes. In absence of grace notes, *g* is zero.

For compatibility reasons, it is possible to write two numbers specifying numerator and denominator instead of the rationals. These forms cannot be mixed, and the two-argument form is disambiguated by the sign of the second argument: if it is positive, it can only be a denominator and not a grace timing.

`ly:make-music` *props* [Funzione]
Make a C++ `Music` object and initialize it with *props*.

This function is for internal use and is only called by `make-music`, which is the preferred interface for creating music objects.

`ly:make-music-function` *signature func* [Funzione]
Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature*'s cdr is a list containing either `ly:music?` predicates or other type predicates. Its car is the syntax function to call.

`ly:make-music-relative!` *music pitch* [Funzione]
Make *music* relative to *pitch*, return final pitch.

`ly:make-output-def` [Funzione]
Make an output definition.

`ly:make-page-label-marker` *label* [Funzione]
Return page marker with label *label*.

`ly:make-page-permission-marker` *symbol permission* [Funzione]
Return page marker with page breaking and turning permissions.

`ly:make-pango-description-string` *chain size* [Funzione]
Make a `PangoFontDescription` string for the property alist *chain* at size *size*.

`ly:make-paper-outputter` *port format* [Funzione]
Create an outputter that evaluates within `output-`*format*, writing to *port*.

`ly:make-pitch` *octave note alter* [Funzione]
octave is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional *alter* is a rational number of 200-cent whole tones for alteration.

`ly:make-prob` *type init rest* [Funzione]
Create a `Prob` object.

`ly:make-scale` *steps* [Funzione]
Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.

`ly:make-score` *music* [Funzione]
Return score with *music* encapsulated in it.

`ly:make-simple-closure` *expr* [Funzione]
Make a simple closure. *expr* should be form of `(`*`func a1 a2 ...`*`)`, and will be invoked as `(`*`func delayed-arg a1 a2 ...`*`)`.

`ly:make-spring` *ideal min-dist* [Funzione]
Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.

`ly:make-stencil` *expr xext yext* [Funzione]
Stencils are device independent output expressions. They carry two pieces of information:

1. A specification of how to print this object. This specification is processed by the output backends, for example '`scm/output-ps.scm`'.
2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use `empty-interval` as its value), it is taken to be empty.

`ly:make-stream-event` *cl proplist* [Funzione]
Create a stream event of class *cl* with the given mutable property list.

`ly:make-undead` *object* [Funzione]
This packages *object* in a manner that keeps it from triggering "Parsed object should be dead" messages.

`ly:make-unpure-pure-container` *unpure pure* [Funzione]
Make an unpure-pure container. *unpure* should be an unpure expression, and *pure* should be a pure expression. If *pure* is omitted, the value of *unpure* will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.

`ly:message` *str rest* [Funzione]
A Scheme callable function to issue the message *str*. The message is formatted with `format` and *rest*.

`ly:minimal-breaking` *pb* [Funzione]
Break (pages and lines) the `Paper_book` object *pb* without looking for optimal spacing: stack as many lines on a page before moving to the next one.

`ly:mm` *num* [Funzione]
num mm.

`ly:module->alist` *mod* [Funzione]
Dump the contents of module *mod* as an alist.

`ly:module-copy` *dest src* [Funzione]
Copy all bindings from module *src* into *dest*.

`ly:modules-lookup` *modules sym def* [Funzione]
Look up *sym* in the list *modules*, returning the first occurence. If not found, return *def* or `#f` if *def* isn't specified.

`ly:moment?` *x* [Funzione]
Is *x* a `Moment` object?

`ly:moment<?` *a b* [Funzione]
Compare two moments.

`ly:moment-add` *a b* [Funzione]
Add two moments.

`ly:moment-div` *a b* [Funzione]
Divide two moments.

`ly:moment-grace` *mom* [Funzione]
Extract grace timing as a rational number from *mom*.

`ly:moment-grace-denominator` *mom* [Funzione]
Extract denominator from grace timing.

`ly:moment-grace-numerator` *mom* [Funzione]
Extract numerator from grace timing.

`ly:moment-main` *mom* [Funzione]
Extract main timing as a rational number from *mom*.

`ly:moment-main-denominator` *mom* [Funzione]
Extract denominator from main timing.

`ly:moment-main-numerator` *mom* [Funzione]
Extract numerator from main timing.

`ly:moment-mod` *a b* [Funzione]
Modulo of two moments.

`ly:moment-mul` *a b* [Funzione]
Multiply two moments.

`ly:moment-sub` *a b* [Funzione]
Subtract two moments.

`ly:music?` *obj* [Funzione]
Is *obj* a music object?

`ly:music-compress` *m factor* [Funzione]
Compress music object *m* by moment *factor*.

`ly:music-deep-copy` *m* [Funzione]
Copy *m* and all sub expressions of *m*. *m* may be an arbitrary type; cons cells and music are copied recursively.

`ly:music-duration-compress` *mus fact* [Funzione]
Compress *mus* by factor *fact*, which is a `Moment`.

`ly:music-duration-length` *mus* [Funzione]
Extract the duration field from *mus* and return the length.

`ly:music-function?` *x* [Funzione]
Is *x* a `music-function`?

`ly:music-function-extract` *x* [Funzione]
Return the Scheme function inside *x*.

`ly:music-function-signature` *x* [Funzione]
Return the function signature inside *x*.

`ly:music-length` *mus* [Funzione]
Get the length of music expression *mus* and return it as a `Moment` object.

`ly:music-list?` *lst* [Funzione]
Is *lst* a list of music objects?

`ly:music-mutable-properties` *mus* [Funzione]
Return an alist containing the mutable properties of *mus*. The immutable properties are not available, since they are constant and initialized by the `make-music` function.

`ly:music-output?` *x* [Funzione]
Is *x* a `Music_output` object?

`ly:music-property` *mus sym val* [Funzione]
Return the value for property *sym* of music expression *mus*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:music-set-property!` *mus sym val* [Funzione]
Set property *sym* in music expression *mus* to *val*.

`ly:music-transpose` *m p* [Funzione]
Transpose *m* such that central C is mapped to *p*. Return *m*.

`ly:note-column-accidentals` *note-column* [Funzione]
Return the `AccidentalPlacement` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-column-dot-column` *note-column* [Funzione]
Return the `DotColumn` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-head::stem-attachment` *font-metric glyph-name* [Funzione]
Get attachment in *font-metric* for attaching a stem to notehead *glyph-name*.

`ly:number->string` *s* [Funzione]
Convert *s* to a string without generating many decimals.

`ly:one-line-breaking` *pb* [Funzione]
Put each score on a single line, and put each line on its own page. The paper-width setting will be modified so that every page will be wider than the widest line.

`ly:optimal-breaking` *pb* [Funzione]
Optimally break (pages and lines) the `Paper_book` object *pb* to minimize badness in bother vertical and horizontal spacing.

`ly:option-usage` *port* [Funzione]
Print `ly:set-option` usage. Optional *port* argumentfor the destination defaults to current output port.

`ly:otf->cff` *otf-file-name* [Funzione]
Convert the contents of an OTF file to a CFF file, returning it as a string.

`ly:otf-font?` *font* [Funzione]
Is *font* an OpenType font?

`ly:otf-font-glyph-info` *font glyph* [Funzione]
Given the font metric *font* of an OpenType font, return the information about named glyph *glyph* (a string).

`ly:otf-font-table-data` *font tag* [Funzione]
Extract a table *tag* from *font*. Return empty string for non-existent *tag*.

`ly:otf-glyph-count` *font* [Funzione]
Return the number of glyphs in *font*.

`ly:otf-glyph-list` *font* [Funzione]
Return a list of glyph names for *font*.

`ly:output-def?` *def* [Funzione]
Is *def* an output definition?

`ly:output-def-clone` *def* [Funzione]
Clone output definition *def*.

`ly:output-def-lookup` *def sym val* [Funzione]
Return the value of *sym* in output definition *def* (e.g., `\paper`). If no value is found, return *val* or `'()` if *val* is undefined.

`ly:output-def-parent` *def* [Funzione]
Return the parent output definition of *def*.

`ly:output-def-scope` *def* [Funzione]
Return the variable scope inside *def*.

`ly:output-def-set-variable!` *def sym val* [Funzione]
Set an output definition *def* variable *sym* to *val*.

`ly:output-description` *output-def* [Funzione]
Return the description of translators in *output-def*.

`ly:output-find-context-def` *output-def context-name* [Funzione]
Return an alist of all context defs (matching *context-name*if given) in *output-def*.

`ly:output-formats` [Funzione]
Formats passed to '`--format`' as a list of strings, used for the output.

`ly:outputter-close` *outputter* [Funzione]
Close port of *outputter*.

`ly:outputter-dump-stencil` *outputter stencil* [Funzione]
Dump stencil *expr* onto *outputter*.

`ly:outputter-dump-string` *outputter str* [Funzione]
Dump *str* onto *outputter*.

`ly:outputter-module` *outputter* [Funzione]
Return output module of *outputter*.

`ly:outputter-output-scheme` *outputter expr* [Funzione]
Eval *expr* in module of *outputter*.

`ly:outputter-port` *outputter* [Funzione]
Return output port for *outputter*.

`ly:page-marker?` *x* [Funzione]
Is *x* a `Page_marker` object?

`ly:page-turn-breaking` *pb* [Funzione]
Optimally break (pages and lines) the `Paper_book` object *pb* such that page turns only happen in specified places, returning its pages.

`ly:pango-font?` *f* [Funzione]
Is *f* a pango font?

`ly:pango-font-physical-fonts` *f* [Funzione]
Return alist of `(ps-name file-name font-index)` lists for Pango font *f*.

`ly:paper-book?` *x* [Funzione]
Is *x* a `Paper_book` object?

`ly:paper-book-header` *pb* [Funzione]
Return the header definition (`\header`) in `Paper_book` object *pb*.

`ly:paper-book-pages` *pb* [Funzione]
Return pages in `Paper_book` object *pb*.

`ly:paper-book-paper` *pb* [Funzione]
Return the paper output definition (`\paper`) in `Paper_book` object *pb*.

`ly:paper-book-performances` *pb* [Funzione]
Return performances in `Paper_book` object *pb*.

`ly:paper-book-scopes` *pb* [Funzione]
Return scopes in `Paper_book` object *pb*.

`ly:paper-book-systems` *pb* [Funzione]
Return systems in `Paper_book` object *pb*.

`ly:paper-fonts` *def* [Funzione]
Return a list containing the fonts from output definition *def* (e.g., `\paper`).

`ly:paper-get-font` *def chain* [Funzione]
Find a font metric in output definition *def* satisfying the font-qualifiers in alist chain *chain*, and return it. (An alist chain is a list of alists, containing grob properties.)

`ly:paper-get-number` *def sym* [Funzione]
Return the value of variable *sym* in output definition *def* as a double.

`ly:paper-outputscale` *def* [Funzione]
Return the output-scale for output definition *def*.

`ly:paper-score-paper-systems` *paper-score* [Funzione]
Return vector of `paper_system` objects from *paper-score*.

`ly:paper-system?` *obj* [Funzione]
Is *obj* a C++ `Prob` object of type `paper-system`?

`ly:paper-system-minimum-distance` *sys1 sys2* [Funzione]
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.

`ly:parse-file` *name* [Funzione]
Parse a single `.ly` file. Upon failure, throw `ly-file-failed` key.

`ly:parse-string-expression` *parser-smob ly-code filename line* [Funzione]
Parse the string *ly-code* with *parser-smob*. Return the contained music expression. *filename* and *line* are optional source indicators.

`ly:parsed-undead-list!` [Funzione]
Return the list of objects that have been found live that should have been dead, and clear that list.

`ly:parser-clear-error` *parser* [Funzione]
Clear the error flag for the parser.

`ly:parser-clone` *parser-smob closures location* [Funzione]
Return a clone of *parser-smob*. An association list of port positions to closures can be specified in *closures* in order to have `$` and `#` interpreted in their original lexical environment. If *location* is a valid location, it becomes the source of all music expressions inside.

`ly:parser-define!` *parser-smob symbol val* [Funzione]
Bind *symbol* to *val* in *parser-smob*'s module.

`ly:parser-error` *parser msg input* [Funzione]
Display an error message and make the parser fail.

`ly:parser-has-error?` *parser* [Funzione]
Does *parser* have an error flag?

`ly:parser-include-string` *parser-smob ly-code* [Funzione]
Include the string *ly-code* into the input stream for *parser-smob*. Can only be used in immediate Scheme expressions (`$` instead of `#`).

`ly:parser-lexer` *parser-smob* [Funzione]
Return the lexer for *parser-smob*.

`ly:parser-lookup` *parser-smob symbol* [Funzione]
Look up *symbol* in *parser-smob*'s module. Return `'()` if not defined.

`ly:parser-output-name` *parser* [Funzione]
Return the base name of the output file.

`ly:parser-parse-string` *parser-smob ly-code* [Funzione]
Parse the string *ly-code* with *parser-smob*. Upon failure, throw `ly-file-failed` key.

`ly:parser-set-note-names` *parser names* [Funzione]
Replace current note names in *parser*. *names* is an alist of symbols. This only has effect if the current mode is notes.

`ly:performance-write` *performance filename* [Funzione]
Write *performance* to *filename*.

`ly:pfb->pfa` *pfb-file-name* [Funzione]
Convert the contents of a Type 1 font in PFB format to PFA format.

`ly:pitch?` *x* [Funzione]
Is *x* a `Pitch` object?

`ly:pitch<?` *p1 p2* [Funzione]
Is *p1* lexicographically smaller than *p2*?

`ly:pitch-alteration` *pp* [Funzione]
Extract the alteration from pitch *pp*.

`ly:pitch-diff` *pitch root* [Funzione]
Return pitch *delta* such that *pitch* transposed by *delta* equals *root*.

`ly:pitch-negate` *p* [Funzione]
Negate *p*.

`ly:pitch-notename` *pp* [Funzione]
Extract the note name from pitch *pp*.

`ly:pitch-octave` *pp* [Funzione]
Extract the octave from pitch *pp*.

`ly:pitch-quartertones` *pp* [Funzione]
Calculate the number of quarter tones of *pp* from middle C.

`ly:pitch-semitones` *pp* [Funzione]
Calculate the number of semitones of *pp* from middle C.

`ly:pitch-steps` *p* [Funzione]
Number of steps counted from middle C of the pitch *p*.

`ly:pitch-tones` *pp* [Funzione]
Calculate the number of tones of *pp* from middle C as a rational number.

`ly:pitch-transpose` *p delta* [Funzione]
Transpose *p* by the amount *delta*, where *delta* is relative to middle C.

`ly:pointer-group-interface::add-grob` *grob sym grob-element* [Funzione]
Add *grob-element* to *grob*'s *sym* grob array.

`ly:position-on-line?` *sg spos* [Funzione]
Return whether *spos* is on a line of the staff associated with the grob *sg* (even on an extender line).

`ly:prob?` *x* [Funzione]
Is *x* a `Prob` object?

`ly:prob-immutable-properties` *prob* [Funzione]
Retrieve an alist of immutable properties.

`ly:prob-mutable-properties` *prob* [Funzione]
Retrieve an alist of mutable properties.

`ly:prob-property` *prob sym val* [Funzione]
Return the value for property *sym* of Prob object *prob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:prob-property?` *obj sym* [Funzione]
Is boolean prop *sym* of *sym* set?

`ly:prob-set-property!` *obj sym value* [Funzione]
Set property *sym* of *obj* to *value*.

`ly:prob-type?` *obj type* [Funzione]
Is *obj* the specified prob-type?

`ly:programming-error` *str rest* [Funzione]
A Scheme callable function to issue the internal warning *str*. The message is formatted with `format` and *rest*.

`ly:progress` *str rest* [Funzione]
A Scheme callable function to print progress *str*. The message is formatted with `format` and *rest*.

`ly:property-lookup-stats` *sym* [Funzione]
Return hash table with a property access corresponding to *sym*. Choices are `prob`, `grob`, and `context`.

`ly:protects` [Funzione]
Return hash of protected objects.

`ly:pt` *num* [Funzione]
num printer points.

`ly:register-stencil-expression` *symbol* [Funzione]
Add *symbol* as head of a stencil expression.

`ly:relative-group-extent` *elements common axis* [Funzione]
Determine the extent of *elements* relative to *common* in the *axis* direction.

`ly:reset-all-fonts` [Funzione]
Forget all about previously loaded fonts.

`ly:round-filled-box` *xext yext blot* [Funzione]
Make a `Stencil` object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.

`ly:round-filled-polygon` *points blot* [Funzione]
Make a `Stencil` object that prints a black polygon with corners at the points defined by *points* (list of coordinate pairs) and roundness *blot*.

`ly:run-translator` *mus output-def* [Funzione]
Process *mus* according to *output-def*. An interpretation context is set up, and *mus* is interpreted with it. The context is returned in its final state.

Optionally, this routine takes an object-key to to uniquely identify the score block containing it.

`ly:score?` *x* [Funzione]
Is *x* a `Score` object?

`ly:score-add-output-def!` *score def* [Funzione]
Add an output definition *def* to *score*.

`ly:score-embedded-format` *score layout* [Funzione]
Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

`ly:score-error?` *score* [Funzione]
Was there an error in the score?

`ly:score-header` *score* [Funzione]
Return score header.

`ly:score-music` *score* [Funzione]
Return score music.

`ly:score-output-defs` *score* [Funzione]
All output definitions in a score.

`ly:score-set-header!` *score module* [Funzione]
Set the score header.

`ly:set-default-scale` *scale* [Funzione]
Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.

`ly:set-grob-modification-callback` *cb* [Funzione]
Specify a procedure that will be called every time LilyPond modifies a grob property. The callback will receive as arguments the grob that is being modified, the name of the C++ file in which the modification was requested, the line number in the C++ file in which the modification was requested, the name of the function in which the modification was requested, the property to be changed, and the new value for the property.

`ly:set-middle-C!` *context* [Funzione]
Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.

`ly:set-option` *var val* [Funzione]
Set a program option.

`ly:set-property-cache-callback` *cb* [Funzione]
Specify a procedure that will be called whenever lilypond calculates a callback function and caches the result. The callback will receive as arguments the grob whose property it is, the name of the property, the name of the callback that calculated the property, and the new (cached) value of the property.

`ly:simple-closure?` *clos* [Funzione]
Is *clos* a simple closure?

`ly:skyline?` *x* [Funzione]
Is *x* a `Skyline` object?

`ly:skyline-empty?` *sky* [Funzione]
Return whether *sky* is empty.

`ly:skyline-pair?` *x* [Funzione]
Is *x* a `Skyline_pair` object?

`ly:slur-score-count` [Funzione]
count number of slur scores.

`ly:smob-protects` [Funzione]
Return LilyPond's internal smob protection list.

`ly:solve-spring-rod-problem` *springs rods length ragged* [Funzione]
Solve a spring and rod problem for *count* objects, that are connected by *count*-1 *springs*, and an arbitrary number of *rods*. *count* is implicitly given by *springs* and *rods*. The *springs* argument has the format `(ideal, inverse_hook)` and *rods* is of the form `(idx1, idx2, distance)`.

length is a number, *ragged* a boolean.

The function returns a list containing the force (positive for stretching, negative for compressing and `#f` for non-satisfied constraints) followed by *spring-count*+1 positions of the objects.

`ly:source-file?` *x* [Funzione]
Is *x* a `Source_file` object?

`ly:spanner?` *g* [Funzione]
Is *g* a spanner object?

`ly:spanner-bound` *spanner dir* [Funzione]
Get one of the bounds of *spanner*. *dir* is `-1` for left, and `1` for right.

`ly:spanner-broken-into` *spanner* [Funzione]
Return broken-into list for *spanner*.

`ly:spanner-set-bound!` *spanner dir item* [Funzione]
Set grob *item* as bound in direction *dir* for *spanner*.

`ly:spawn` *command rest* [Funzione]
Simple interface to g_spawn_sync *str*. The error is formatted with `format` and *rest*.

`ly:spring?` *x* [Funzione]
Is *x* a `Spring` object?

`ly:spring-set-inverse-compress-strength!` *spring strength* [Funzione]
Set the inverse compress *strength* of *spring*.

`ly:spring-set-inverse-stretch-strength!` *spring strength* [Funzione]
Set the inverse stretch *strength* of *spring*.

`ly:staff-symbol-line-thickness` *grob* [Funzione]
Returns the `line-thickness` of the staff associated with *grob*.

`ly:staff-symbol-staff-radius` *grob* [Funzione]
Returns the radius of the staff associated with *grob*.

`ly:staff-symbol-staff-space` *grob* [Funzione]
Returns the `staff-space` of the staff associated with *grob*.

`ly:start-environment` [Funzione]
Return the environment (a list of strings) that was in effect at program start.

`ly:stderr-redirect` *file-name mode* [Funzione]
Redirect stderr to *file-name*, opened with *mode*.

`ly:stencil?` *x* [Funzione]
Is *x* a `Stencil` object?

`ly:stencil-add` *args* [Funzione]
Combine stencils. Takes any number of arguments.

`ly:stencil-aligned-to` *stil axis dir* [Funzione]
Align *stil* using its own extents. *dir* is a number. `-1` and `1` are left and right, respectively. Other values are interpolated (so `0` means the center).

`ly:stencil-combine-at-edge` *first axis direction second padding* [Funzione]
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`.

`ly:stencil-empty?` *stil axis* [Funzione]
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.

`ly:stencil-expr` *stil* [Funzione]
Return the expression of *stil*.

`ly:stencil-extent` *stil axis* [Funzione]
Return a pair of numbers signifying the extent of *stil* in *axis* direction (`0` or `1` for x and y axis, respectively).

`ly:stencil-fonts` *s* [Funzione]
Analyze *s*, and return a list of fonts used in *s*.

`ly:stencil-in-color` *stc r g b* [Funzione]
Put *stc* in a different color.

`ly:stencil-rotate` *stil angle x y* [Funzione]
Return a stencil *stil* rotated *angle* degrees around the relative offset (*x*, *y*). E.g., an offset of (-1, 1) will rotate the stencil around the left upper corner.

`ly:stencil-rotate-absolute` *stil angle x y* [Funzione]
Return a stencil *stil* rotated *angle* degrees around point (*x*, *y*), given in absolute coordinates.

`ly:stencil-scale` *stil x y* [Funzione]
Scale *stil* using the horizontal and vertical scaling factors *x* and *y*.

`ly:stencil-stack` *first axis direction second padding mindist* [Funzione]
Construct a stencil by stacking *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`. As opposed to `ly:stencil-combine-at-edge`, metrics are suited for successively accumulating lines of stencils. Also, *second* stencil is drawn last.

If *mindist* is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing, *padding* and *mindist* do not apply.

`ly:stencil-translate` *stil offset* [Funzione]
Return a *stil*, but translated by *offset* (a pair of numbers).

`ly:stencil-translate-axis` *stil amount axis* [Funzione]
Return a copy of *stil* but translated by *amount* in *axis* direction.

`ly:stream-event?` *obj* [Funzione]
Is *obj* a `Stream_event` object?

`ly:string-percent-encode` *str* [Funzione]
Encode all characters in string *str* with hexadecimal percent escape sequences, with the following exceptions: characters `-`, `.`, `/`, and `_`; and characters in ranges `0-9`, `A-Z`, and `a-z`.

`ly:string-substitute` *a b s* [Funzione]
Replace string *a* by string *b* in string *s*.

`ly:system-font-load` *name* [Funzione]
Load the OpenType system font '*name*`.otf`'. Fonts loaded with this command must contain three additional SFNT font tables called `LILC`, `LILF`, and `LILY`, needed for typesetting musical elements. Currently, only the Emmentaler and the Emmentaler-Brace fonts fulfill these requirements.

Note that only `ly:font-get-glyph` and derived code (like `\lookup`) can access glyphs from the system fonts; text strings are handled exclusively via the Pango interface.

`ly:text-interface::interpret-markup` [Funzione]
Convert a text markup into a stencil. Takes three arguments, *layout*, *props*, and *markup*.

layout is a `\layout` block; it may be obtained from a grob with `ly:grob-layout`. *props* is an alist chain, i.e. a list of alists. This is typically obtained with `(ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults))`. *markup* is the markup text to be processed.

`ly:translate-cpp-warning-scheme` *str* [Funzione]
Translates a string in C++ printf format and modifies it to use it for scheme formatting.

`ly:translator?` *x* [Funzione]
Is *x* a `Translator` object?

`ly:translator-context` *trans* [Funzione]
Return the context of the translator object *trans*.

`ly:translator-description` *me* [Funzione]
Return an alist of properties of translator *me*.

`ly:translator-group?` *x* [Funzione]
Is *x* a `Translator_group` object?

`ly:translator-name` *trans* [Funzione]
Return the type name of the translator object *trans*. The name is a symbol.

`ly:transpose-key-alist` *l pit* [Funzione]
Make a new key alist of *l* transposed by pitch *pit*.

`ly:truncate-list!` *lst i* [Funzione]
Take at most the first *i* of list *lst*.

`ly:ttf->pfa` *ttf-file-name idx* [Funzione]
Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:ttf-ps-name` *ttf-file-name idx* [Funzione]
Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:undead?` *x* [Funzione]
Is *x* a `Undead` object?

`ly:unit` [Funzione]
Return the unit used for lengths as a string.

`ly:unpure-pure-container?` *clos* [Funzione]
Is *clos* an unpure pure container?

`ly:unpure-pure-container-pure-part` *pc* [Funzione]
Return the pure part of *pc*.

`ly:unpure-pure-container-unpure-part` *pc* [Funzione]
Return the unpure part of *pc*.

`ly:usage` [Funzione]
Print usage message.

`ly:verbose-output?` [Funzione]
Was verbose output requested, i.e. loglevel at least `DEBUG`?

`ly:version` [Funzione]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.
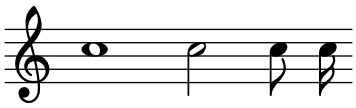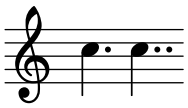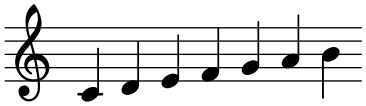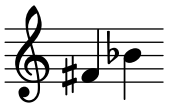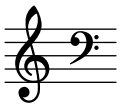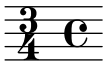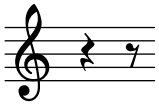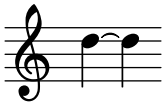
`ly:warning` *str rest* [Funzione]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.

`ly:warning-located` *location str rest* [Funzione]
A Scheme callable function to issue the warning *str* at the specified location in an input file. The message is formatted with `format` and *rest*.
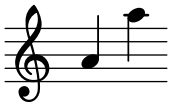
`ly:wide-char->utf-8` *wc* [Funzione]

Encode the Unicode codepoint *wc*, an integer, as UTF-8.
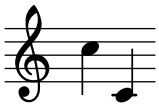
Appendice B Cheat sheet

Syntax	Description	Example
`1 2 8 16`	durations	
`c4. c4..`	augmentation dots	
`c d e f g a b`	scale	
`fis bes`	alteration	
`\clef treble \clef bass`	clefs	
`\time 3/4 \time 4/4`	time signature	
`r4 r8`	rest	
`d ~ d`	tie	
`\key es \major`	key signature	
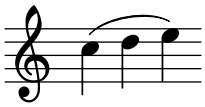
note`'`	raise octave
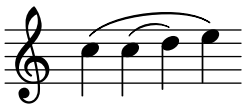


note`,`	lower octave
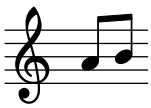


`c( d e)`	slur



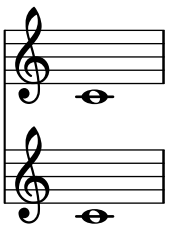
`c\( c( d) e\)`	phrasing slur
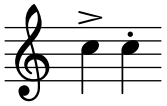


`a8[ b]`	beam
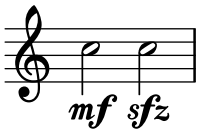


`<< \new Staff ... >>`	more staves



`c-> c-.`	articulations
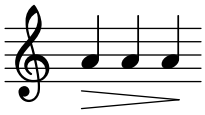


`c2\mf c\sfz`	dynamics
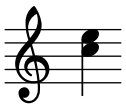


`a\< a a\!`	crescendo
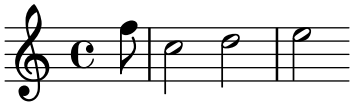


`a\> a a\!` decrescendo
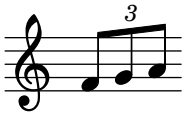


`< >` chord
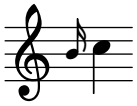


`\partial 8` pickup / upbeat



`\tuplet 3/2 {f g a}` triplets



`\grace` grace notes

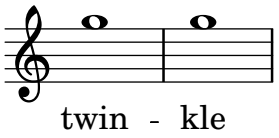


`\lyricmode { twinkle }` entering lyrics twinkle
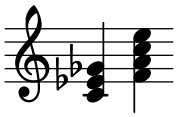
`\new Lyrics` printing lyrics

twinkle

`twin -- kle` lyric hyphen
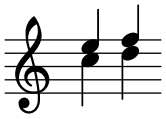


`\chordmode { c:dim f:maj7 }` chords



`\context ChordNames` printing chord names

C° F△

`<<{e f} \\ {c d}>>` polyphony



`s4 s8 s16`	spacer rests

Appendice C GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

   The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendice D Indice dei comandi di LilyPond

Questo indice elenca tutti i comandi e le parole chiave di LilyPond con dei collegamenti alle sezioni del manuale che descrivono il loro uso. Ogni collegamento è composto da due parti. La prima parte porta al punto esatto del manuale in cui compaiono il comando o la parola chiave; la seconda parte porta all'inizio della sezione del manuale in cui compaiono il comando o la parola.

!

! 6

"

"|" 104

'

' 1

,

, ... 1

-

- 115

.

. 43

/

/ 397
/+ 397

:

: 158

<

< 159
<...> 159

=

= 9

>

> 159

?

? 6

[

[..... 89

]

] 89

^

^ 396

_

_ 255

\

\! 118
\(..... 129
\) 129
\< 118
\> 118
\abs-fontsize 232, 653
\absolute 2
\accent 115
\accepts 568, 569, 570
\acciaccatura 107
\accidentalStyle 26
\addChordShape 355
\addInstrumentDefinition 201, 209
\addlyrics 249, 251, 252
\addQuote 202
\aeolian 21
\afterGrace 108
\aikenHeads 38
\aikenHeadsMinor 39
\alias 568
\allowPageTurn 519
\alterBroken 609
\alternative 143
\appendToTag 483
\appoggiatura 107
\arpeggio 137
\arpeggioArrowDown 137
\arpeggioArrowUp 137
\arpeggioBracket 137
\arpeggioNormal 137
\arpeggioParenthesis 137
\arpeggioParenthesisDashed 137
\arrow-head 239
\arrow-head 677
\ascendens 429, 435
\auctum 429, 435
\augmentum 435
\auto-footnote 697
\autoBeamOff 78, 315
\autoBeamOn 78
\autochange 313
\backslashed-digit 697

\balloonGrobText ... 218
\balloonLengthOff ... 218
\balloonLengthOn ... 218
\balloonText ... 218
\bar ... 93, 99
\barNumberCheck ... 105
\beam ... 677
\bendAfter ... 132
\bold ... 231, 654
\book ... 451, 454
\bookOutputName ... 453
\bookOutputSuffix ... 453
\bookpart ... 452, 454, 517
\box ... 237, 654
\bracket ... 124, 237, 677
\break ... 517
\breathe ... 130
\breve ... 43, 54
\cadenzaOff ... 70
\cadenzaOn ... 70
\caesura ... 427
\caps ... 654
\cavum ... 429, 435
\center-align ... 234, 662
\center-column ... 236, 662
\change ... 312
\char ... 698
\chordmode ... 5, 13, 353
\chordRepeats ... 328
\chords ... 399
\circle ... 237, 677
\clef ... 17
\cm ... 586
\coda ... 115
\column ... 236, 663
\column-lines ... 703
\combine ... 239
\combine ... 663
\compoundMeter ... 74
\compressFullBarRests ... 58, 59
\concat ... 663
\consists ... 568
\context ... 555, 563
\cr ... 118
\cresc ... 119
\crescHairpin ... 119
\crescTextCresc ... 119
\crossStaff ... 315
\cueClef ... 205
\cueDuring ... 205
\cueDuringWithClef ... 205
\customTabClef ... 684
\decr ... 118
\decresc ... 119
\defaultchild ... 571
\defaultTimeSignature ... 62
\defineBarLine ... 97
\deminutum ... 429, 435
\denies ... 568, 569, 570
\descendens ... 429, 435
\dim ... 119
\dimHairpin ... 119
\dimTextDecr ... 119
\dimTextDecresc ... 119
\dimTextDim ... 119
\dir-column ... 664
\discant ... 693
\displayLilyMusic ... 500
\divisioMaior ... 427
\divisioMaxima ... 427
\divisioMinima ... 427
\dorian ... 21
\dotsDown ... 44
\dotsNeutral ... 44
\dotsUp ... 44
\doubleflat ... 684
\doublesharp ... 684
\downbow ... 115, 321
\downmordent ... 115
\downprall ... 115
\draw-circle ... 239
\draw-circle ... 678
\draw-dashed-line ... 678
\draw-dotted-line ... 678
\draw-hline ... 679
\draw-line ... 239
\draw-line ... 679
\drummode ... 181
\dynamic ... 124, 654
\dynamicDown ... 120
\dynamicNeutral ... 120
\dynamicUp ... 120
\easyHeadsOff ... 37
\easyHeadsOn ... 37
\ellipse ... 679
\epsfile ... 239
\epsfile ... 680
\espressivo ... 115, 119
\expandFullBarRests ... 58, 59
\eyeglasses ... 698
\f ... 118
\featherDurations ... 92
\fermata ... 115, 684
\fermataMarkup ... 58, 59, 115
\ff ... 118
\fff ... 118
\ffff ... 118
\fffff ... 118
\fill-line ... 236, 664
\fill-with-pattern ... 665
\filled-box ... 239
\filled-box ... 680
\finalis ... 427
\finger ... 212, 655
\flageolet ... 115
\flat ... 685
\flexa ... 435
\fontCaps ... 655
\fontsize ... 232, 655
\footnote ... 468, 698
\fp ... 118
\fraction ... 698
\freeBass ... 693
\frenchChords ... 402
\fret-diagram ... 343, 689
\fret-diagram-terse ... 345, 690
\fret-diagram-verbose ... 347, 690
\fromproperty ... 698
\funkHeads ... 38
\funkHeadsMinor ... 39

\general-align 235, 665
\germanChords 402
\glissando 133
\grace 107
\halfopen 115
\halign 234, 666
\harmonic 322, 330
\harmonicByFret 330
\harmonicByRatio 330
\harmonicsOff 322
\harmonicsOn 322
\harp-pedal 691
\hbracket 237, 680
\hcenter-in 667
\header 454
\hide 592
\hideKeySignature 382
\hideNotes 214
\hideSplitTiedTabNotes 329
\hideStaffSwitch 315
\hspace 668
\huge 210
\huge 233
\huge 655
\improvisationOff 41, 76
\improvisationOn 41, 76
\in 586
\inclinatum 429, 435
\include 480
\inStaffSegno 146
\instrumentSwitch 201
\inversion 13
\ionian 21
\italianChords 402
\italic 231, 655
\justified-lines 242, 703
\justify 236, 669
\justify-field 668
\justify-string 669
\keepWithTag 483
\key 20, 39
\killCues 209
\label 478
\laissezVibrer 51
\large 210
\large 233
\large 656
\larger 232
\larger 233
\larger 656
\layout 454, 512
\layout 553
\layout 563
\left-align 234, 670
\left-brace 699
\left-column 670
\lheel 115
\line 670
\linea 429, 435
\lineprall 115
\locrian 21
\longa 43, 54
\longfermata 115
\lookup 699
\lower 234, 671
\ltoe 115
\lydian 21
\lyricmode 248, 249
\lyricsto 249, 251
\magnify 232, 656
\major 21
\makeClusters 164
\map-markup-commands 704
\marcato 115
\mark 105, 226
\markalphabet 699
\markLengthOff 67, 227
\markLengthOn 67, 227
\markletter 700
\markup 226, 228, 229, 230
\markuplist 229, 242, 243
\maxima 43, 54
\medium 656
\melisma 255
\melismaEnd 255
\mergeDifferentlyDottedOff 168
\mergeDifferentlyDottedOn 168
\mergeDifferentlyHeadedOff 168
\mergeDifferentlyHeadedOn 168
\mf 118
\midi 454
\midi 553
\minor 21
\mixolydian 21
\mm 586
\modalInversion 15
\modalTranspose 14
\mordent 115
\mp 118
\musicglyph 106, 685
\name 568
\natural 685
\new 555
\newSpacingSection 540
\noBeam 90
\noBreak 517
\noPageBreak 518
\noPageTurn 519
\normal-size-sub 657
\normal-size-super 232, 657
\normal-text 657
\normalsize 210
\normalsize 233
\normalsize 658
\note 686
\note-by-number 685
\null 234, 700
\number 658
\numericTimeSignature 62
\octaveCheck 9
\omit 592
\on-the-fly 467, 700
\once 578
\oneVoice 164
\open 115, 321
\oriscus 429, 435
\ottava 23
\oval 680
\override 577, 581, 700
\override-lines 704

\overrideProperty ... 581
\overrideTimeSignatureSettings ... 62
\p ... 118
\pad-around ... 238, 671
\pad-markup ... 238, 671
\pad-to-box ... 238, 672
\pad-x ... 238, 672
\page-link ... 701
\page-ref ... 478, 701
\pageBreak ... 518
\pageTurn ... 519
\paper ... 454, 503
\parallelMusic ... 178
\parenthesize ... 216, 681
\partcombine ... 173, 276
\partcombineApart ... 174
\partcombineAutomatic ... 174
\partcombineChords ... 174
\partcombineSoloI ... 174
\partcombineSoloII ... 174
\partcombineUnisono ... 174
\partial ... 69, 143, 144
\path ... 681
\pattern ... 701
\pes ... 435
\phrasingSlurDashed ... 129
\phrasingSlurDashPattern ... 129
\phrasingSlurDotted ... 129
\phrasingSlurDown ... 129
\phrasingSlurHalfDashed ... 129
\phrasingSlurHalfSolid ... 129
\phrasingSlurNeutral ... 129
\phrasingSlurSolid ... 129
\phrasingSlurUp ... 129
\phrygian ... 21
\pitchedTrill ... 141
\portato ... 115
\postscript ... 239
\postscript ... 682
\powerChords ... 368
\pp ... 118
\ppp ... 118
\pppp ... 118
\ppppp ... 118
\prall ... 115
\pralldown ... 115
\prallmordent ... 115
\prallprall ... 115
\prallup ... 115
\predefinedFretboardsOff ... 362
\predefinedFretboardsOn ... 362
\property-recursive ... 701
\pt ... 586
\pushToTag ... 483
\put-adjacent ... 672
\quilisma ... 429, 435
\quoteDuring ... 202, 205
\raise ... 234, 672
\relative ... 2, 5, 13, 314
\RemoveEmptyStaves ... 195, 197
\removeWithTag ... 483
\repeat ... 143
\repeat percent ... 155
\repeat tremolo ... 157
\repeatTie ... 51, 146, 272
\replace ... 658
\rest ... 54, 687
\rest-by-number ... 686
\retrograde ... 14
\reverseturn ... 115
\revert ... 578
\revertTimeSignatureSettings ... 64
\rfz ... 118
\rheel ... 115
\right-align ... 234, 673
\right-brace ... 701
\right-column ... 673
\rightHandFinger ... 364
\roman ... 658
\rotate ... 673
\rounded-box ... 237, 683
\rtoe ... 115
\sacredHarpHeads ... 38
\sacredHarpHeadsMinor ... 39
\sans ... 659
\scale ... 683
\scaleDurations ... 50, 72
\score ... 450, 454, 687
\segno ... 115
\semiflat ... 688
\semiGermanChords ... 402
\semisharp ... 688
\sesquiflat ... 688
\sesquisharp ... 689
\set ... 81, 575, 581
\sf ... 118
\sff ... 118
\sfz ... 118
\shape ... 606
\sharp ... 689
\shiftOff ... 168
\shiftOn ... 168
\shiftOnn ... 168
\shiftOnnn ... 168
\shortfermata ... 115
\showKeySignature ... 382
\showStaffSwitch ... 315
\signumcongruentiae ... 115
\simple ... 659
\skip ... 56, 271
\slashed-digit ... 701
\slashedGrace ... 107
\slurDashed ... 126
\slurDashPattern ... 127
\slurDotted ... 126
\slurDown ... 126
\slurHalfDashed ... 127
\slurHalfSolid ... 127
\slurNeutral ... 126
\slurSolid ... 126
\slurUp ... 127
\small ... 210
\small ... 233
\small ... 659
\smallCaps ... 659
\smaller ... 232
\smaller ... 233
\smaller ... 660
\snappizzicato ... 115
\sostenutoOff ... 317

\sostenutoOn ... 317
\southernHarmonyHeads ... 38
\southernHarmonyHeadsMinor ... 39
\sp ... 118
\spp ... 118
\staccatissimo ... 115
\staccato ... 115
\startGroup ... 221
\startStaff ... 188, 192
\startTrillSpan ... 140
\stdBass ... 694
\stdBassIV ... 695
\stdBassV ... 695
\stdBassVI ... 696
\stemDown ... 217
\stemNeutral ... 217
\stemUp ... 217
\stencil ... 702
\stopGroup ... 221
\stopped ... 115
\stopStaff ... 188, 192, 195
\stopTrillSpan ... 140
\storePredefinedDiagram ... 355
\stringTuning ... 340
\stropha ... 429, 435
\strut ... 702
\sub ... 232, 660
\super ... 232, 660
\sustainOff ... 317
\sustainOn ... 317
\tabChordRepeats ... 328
\tabFullNotation ... 327
\table-of-contents ... 480, 704
\tag ... 483
\taor ... 382
\teeny ... 210
\teeny ... 233
\teeny ... 661
\tempo ... 66
\tenuto ... 115
\text ... 661
\textLengthOff ... 59, 223
\textLengthOn ... 59, 223
\textSpannerDown ... 224
\textSpannerNeutral ... 224
\textSpannerUp ... 224
\thumb ... 115, 212
\tied-lyric ... 689
\tieDashed ... 52
\tieDotted ... 52
\tieDown ... 52
\tieNeutral ... 52
\tieSolid ... 52
\tieUp ... 52
\time ... 61, 81
\tiny ... 210
\tiny ... 233
\tiny ... 661
\tocItem ... 480
\translate ... 235, 674
\translate-scaled ... 235, 674
\transparent ... 702
\transpose ... 5, 10, 13
\transposedCueDuring ... 208
\transposition ... 24, 202
\treCorde ... 317
\triangle ... 239
\triangle ... 684
\trill ... 115, 140
\tuplet ... 45
\tuplet ... 72
\tupletDown ... 45
\tupletNeutral ... 45
\tupletUp ... 45
\turn ... 115
\tweak ... 579, 581
\type ... 568
\typewriter ... 661
\unaCorda ... 317
\underline ... 231, 661
\unfoldRepeats ... 495
\unHideNotes ... 214
\unset ... 576
\upbow ... 115, 321
\upmordent ... 115
\upprall ... 115
\upright ... 662
\varcoda ... 115
\vcenter ... 674
\verbatim-file ... 702
\verylongfermata ... 115
\virga ... 429, 435
\virgula ... 427
\voiceFourStyle ... 168
\voiceNeutralStyle ... 168
\voiceOne ... 164
\voiceOne ... \voiceFour ... 164
\voiceOneStyle ... 168
\voiceThreeStyle ... 168
\voiceTwoStyle ... 168
\void ... 501
\vspace ... 674
\walkerHeads ... 38
\walkerHeadsMinor ... 39
\whiteout ... 702
\with ... 561, 566
\with-color ... 215, 703
\with-dimensions ... 703
\with-link ... 703
\with-url ... 684
\woodwind-diagram ... 692
\wordwrap ... 236, 675
\wordwrap-field ... 675
\wordwrap-internal ... 704
\wordwrap-lines ... 242, 704
\wordwrap-string ... 676
\wordwrap-string-internal ... 704

|

| ... 104

~

~ ... 50

A

absolute ... 2, 742

accepts 568
acciaccatura 742
accidentalStyle 742
addChordShape 355, 742
addInstrumentDefinition 201, 209, 742
additionalPitchPrefix 401
addQuote 202, 742
aeolian 21
afterGrace 108, 743
aikenHeads 38
aikenHeadsMinor 39
alias 568
alignAboveContext 571
alignBelowContext 270, 571
allowPageTurn 743
allowVoltaHook 743
alterBroken 743
annotate-spacing 549
appendToTag 743
applyContext 743
applyMusic 743
applyOutput 743
appoggiatura 743
arpeggio 137
arpeggioArrowDown 137
arpeggioArrowUp 137
arpeggioBracket 137
arpeggioNormal 137
arpeggioParenthesis 137
arpeggioParenthesisDashed 137
arrow-head 239
assertBeamQuant 743
assertBeamSlope 743
aug 394
auto-first-page-number 511
autoBeaming 81
autoBeaming 553
autoBeamOff 78
autoBeamOn 78
autochange 313, 743

B

Balloon_engraver 218
balloonGrobText 218, 743
balloonLengthOff 218
balloonLengthOn 218
balloonText 218, 743
banjo-c-tuning 370
banjo-modal-tuning 370
banjo-open-d-tuning 370
banjo-open-dm-tuning 370
bar 93, 99, 743
barCheckSynchronize 104
BarNumber 100
barNumberCheck 105, 743
barNumberVisibility 100
bartype 99
base-shortest-duration 539
baseMoment 81
beamExceptions 81
beatStructure 81
bendAfter 132, 743
binding-offset 508
blank-after-score-page-penalty 510
blank-last-page-penalty 510
blank-page-penalty 510
bold 231
bookOutputName 743
bookOutputSuffix 743
bookTitleMarkup 464
bottom-margin 505
box 237
bracket 124, 237, 317
breakable 79
breathe 130, 744
breve 43, 54

C

cadenzaOff 70
cadenzaOn 70
center-align 234
center-column 236
change 312
check-consistency 508
chordChanges 399
chordmode 5, 13, 353
chordNameExceptions 402
chordNameLowercaseMinor 400
ChordNames 353
chordNameSeparator 401
chordNoteNamer 401
chordPrefixSpacer 402
chordRepeats 744
chordRootNamer 401
circle 237
clef 17, 744
color 215
column 236
combine 239
common-shortest-duration 539
Completion_heads_engraver 75
Completion_rest_engraver 75
compoundMeter 744
compressFullBarRests 58, 59
consists 568
controlpitch 9
cr 118
cresc 119
crescHairpin 119
crescTextCresc 119
cross 35
crossStaff 744
cueClef 205, 744
cueClefUnset 744
cueDuring 205, 744
cueDuringWithClef 205, 744
currentBarNumber 100, 114

D

deadNote 744
decr 118
decresc 119
default 26, 28
default-staff-staff-spacing 523
defaultBarType 99

defaultNoteHeads 744
defaultTimeSignature 62
defineBarLine 97, 744
denies 568
dim 119
dim 394
dimHairpin 119
dimTextDecr 119
dimTextDecresc 119
dimTextDim 119
displayLilyMusic 744
displayMusic 744
displayScheme 744
dodecaphonic 31
dorian 21
dotsDown 44
dotsNeutral 44
dotsUp 44
draw-circle 239
draw-line 239
drummode 181
DrumStaff 181
dynamic 124
dynamicDown 120
DynamicLineSpanner 120
dynamicNeutral 120
dynamicUp 120

E

easyHeadsOff 37
easyHeadsOn 37
endSpanners 744
epsfile 239
espressivo 119
eventChords 745
expandFullBarRests 58, 59
extra-offset 523

F

f 118
featherDurations 92, 745
fermataMarkup 58, 59
ff 118
fff 118
ffff 118
fffff 118
fill-line 236
filled-box 239
finger 212, 745
first-page-number 511
followVoice 315
font-interface 211, 243
font-size 210, 211
fontsize 232
fontSize 210
footnote 745
forget 31
four-string-banjo 370
fp 118
fret-diagram 343
fret-diagram-interface 348
fret-diagram-terse 345
fret-diagram-verbose 347
FretBoards 351
funkHeads 38
funkHeadsMinor 39

G

general-align 235
glissando 133
grace 745
GregorianTranscriptionStaff 181
Grid_line_span_engraver 219
Grid_point_engraver 219
gridInterval 219
grobdescriptions 745
grow-direction 92

H

halign 234
harmonicByFret 745
harmonicByRatio 745
harmonicNote 745
harmonicsOn 745
hbracket 237
hide 745
hideKeySignature 382
hideNotes 214
hideStaffSwitch 315
horizontal-shift 509
Horizontal_bracket_engraver 221
huge 210
huge 233

I

improvisationOff 41, 76
improvisationOn 41, 76
indent 200, 509, 542
inner-margin 508
inStaffSegno 745
instrumentSwitch 201, 746
inversion 746
ionian 21
italic 231

J

justified-lines 242
justify 236

K

keepWithTag 746
key 20, 39, 746
killCues 209, 746

L

label 746
laissezVibrer 51
language 746
languageRestore 746
languageSaveAndChange 746

large ... 210
large ... 233
larger ... 232
larger ... 233
last-bottom-spacing ... 507
layout file ... 514
left-align ... 234
left-margin ... 507
line-width ... 507, 542
locrian ... 21
longa ... 43, 54
lower ... 234
ly:minimal-breaking ... 519
ly:one-line-breaking ... 519
ly:optimal-breaking ... 518
ly:page-turn-breaking ... 518
lydian ... 21

M

m ... 394
magnify ... 232
magstep ... 210, 586
maj ... 394
major ... 21
major seven symbols ... 402
majorSevenSymbol ... 401
make-dynamic-script ... 124
make-pango-font-tree ... 245
makeClusters ... 164, 746
makeDefaultStringTuning ... 746
mark ... 105, 226, 746
markLengthOff ... 67, 227
markLengthOn ... 67, 227
markup ... 226, 228, 229, 230
markup-markup-spacing ... 507
markup-system-spacing ... 506
markuplist ... 229, 242, 243
max-systems-per-page ... 509
maxima ... 43, 54
measureLength ... 81, 114
measurePosition ... 69, 114
MensuralStaff ... 181
mergeDifferentlyDottedOff ... 168
mergeDifferentlyDottedOn ... 168
mergeDifferentlyHeadedOff ... 168
mergeDifferentlyHeadedOn ... 168
mf ... 118
min-systems-per-page ... 509
minimum-Y-extent ... 523
minimumFret ... 327, 363
minimumPageTurnLength ... 518
minimumRepeatLengthForPageTurn ... 519
minor ... 21
minorChordModifier ... 402
mixed ... 317
mixolydian ... 21
modalInversion ... 15, 746
modalTranspose ... 14, 746
modern ... 28
modern-cautionary ... 29
modern-voice ... 29
modern-voice-cautionary ... 29
mp ... 118
MultiMeasureRestText ... 58
musicglyph ... 106
musicMap ... 746

N

name ... 568
neo-modern ... 30
neo-modern-cautionary ... 30
neo-modern-voice ... 30
neo-modern-voice-cautionary ... 31
no-reset ... 31
noBeam ... 90
nonstaff-nonstaff-spacing ... 523
nonstaff-relatedstaff-spacing ... 523
nonstaff-unrelatedstaff-spacing ... 523
noPageBreak ... 746
noPageTurn ... 746
normal-size-super ... 232
normalsize ... 210
normalsize ... 233
Note_heads_engraver ... 75
null ... 234
numericTimeSignature ... 62

O

octaveCheck ... 9, 746
offset ... 746
omit ... 747
once ... 747
oneVoice ... 164
ottava ... 23, 747
outer-margin ... 508
outside-staff-horizontal-padding ... 537
outside-staff-padding ... 537
outside-staff-priority ... 537
overrideProperty ... 747
overrideTimeSignatureSettings ... 747

P

p ... 118
pad-around ... 238
pad-markup ... 238
pad-to-box ... 238
pad-x ... 238
page-breaking ... 510
page-breaking-system-system-spacing ... 510
page-count ... 510
page-spacing-weight ... 511
pageBreak ... 747
pageTurn ... 747
palmMute ... 747
palmMuteOn ... 747
paper-height ... 505
paper-width ... 507
parallelMusic ... 178, 747
parenthesize ... 216, 747
partcombine ... 173, 747
partcombineApart ... 174
partcombineAutomatic ... 174
partcombineChords ... 174
partcombineDown ... 748

partcombineForce ... 748
partcombineSoloI ... 174
partcombineSoloII ... 174
partcombineUnisono ... 174
partcombineUp ... 748
partial ... 69, 748
pedalSustainStyle ... 317
percent ... 155
phrasingSlurDashed ... 129
phrasingSlurDashPattern ... 129, 748
phrasingSlurDotted ... 129
phrasingSlurDown ... 129
phrasingSlurHalfDashed ... 129
phrasingSlurHalfSolid ... 129
phrasingSlurNeutral ... 129
phrasingSlurSolid ... 129
phrasingSlurUp ... 129
phrygian ... 21
piano ... 29
piano-cautionary ... 30
PianoStaff ... 311, 313
Pitch_squash_engraver ... 76
pitchedTrill ... 141, 748
pointAndClickOff ... 748
pointAndClickOn ... 748
pointAndClickTypes ... 748
postscript ... 239
powerChords ... 368
pp ... 118
ppp ... 118
pppp ... 118
ppppp ... 118
predefinedFretboardsOff ... 362
predefinedFretboardsOn ... 362
print-all-headers ... 511
print-first-page-number ... 511
print-page-number ... 511
pushToTag ... 748

Q

quotedCueEventTypes ... 204
quotedEventTypes ... 204
quoteDuring ... 202, 205, 748

R

r ... 54
R ... 57
ragged-bottom ... 505
ragged-last ... 508, 542
ragged-last-bottom ... 505
ragged-right ... 508, 542
raise ... 234
relative ... 2, 5, 13, 314, 748
removeWithTag ... 748
repeatCommands ... 151
repeatTie ... 51
resetRelativeOctave ... 748
rest ... 54
restrainOpenStrings ... 327
retrograde ... 14, 748
revertTimeSignatureSettings ... 748
rfz ... 118
rgb-color ... 215
RhythmicStaff ... 181
right-align ... 234
right-margin ... 508
rightHandFinger ... 364
rightHandFinger ... 749
rounded-box ... 237

S

s ... 56
sacredHarpHeads ... 38
sacredHarpHeadsMinor ... 39
scaleDurations ... 50, 72, 749
score-markup-spacing ... 506
score-system-spacing ... 506
scoreTitleMarkup ... 464
self-alignment-X ... 523
set ... 81
set-octavation ... 23
settingsFrom ... 749
sf ... 118
sff ... 118
sfz ... 118
shape ... 749
shiftDurations ... 749
shiftOff ... 168
shiftOn ... 168
shiftOnn ... 168
shiftOnnn ... 168
short-indent ... 200, 509
show-available-fonts ... 245
showFirstLength ... 489
showKeySignature ... 382
showLastLength ... 489
showStaffSwitch ... 315
single ... 749
skip ... 56, 749
skipTypesetting ... 489
slashChordSeparator ... 402
slashedGrace ... 749
slurDashed ... 126
slurDashPattern ... 127, 749
slurDotted ... 126
slurDown ... 126
slurHalfDashed ... 127
slurHalfSolid ... 127
slurNeutral ... 126
slurSolid ... 126
slurUp ... 127
small ... 210
small ... 233
smaller ... 232
smaller ... 233
sostenutoOff ... 317
sostenutoOn ... 317
southernHarmonyHeads ... 38
southernHarmonyHeadsMinor ... 39
sp ... 118
spacing ... 539
spacingTweaks ... 749
Span_stem_engraver ... 315
spp ... 118
staff-affinity ... 523
staff-staff-spacing ... 523

Staff.midiInstrument 494
Staff_symbol_engraver 195
staffgroup-staff-spacing 523
start-repeat 151
startGroup 221
startStaff 188, 192
startTrillSpan 140
Stem 315
stem-spacing-correction 539
stemDown 217
stemLeftBeamCount 90
stemNeutral 217
stemRightBeamCount 90
stemUp 217
stopGroup 221
stopStaff 188, 192, 195
stopTrillSpan 140
storePredefinedDiagram 355, 749
stringTuning 340, 749
stringTunings 339, 351
styledNoteHeads 749
sub 232
suggestAccidentals 422
super 232
sus 396
sustainOff 317
sustainOn 317
system-count 509
system-separator-markup 511
system-system-spacing 507
systems-per-page 509

T

tabChordRepeats 749
tabChordRepetition 749
TabStaff 181, 326
TabVoice 326
tag 750
taor 382
teaching 31
teeny 210
teeny 233
tempo 66
temporary 750
text 317
textLengthOff 59, 223
textLengthOn 59, 223
textSpannerDown 224
textSpannerNeutral 224
textSpannerUp 224
thumb 212
tieDashed 52
tieDashPattern 750
tieDotted 52
tieDown 52
tieNeutral 52
tieSolid 52
tieUp 52
time 61, 81, 750
times 750
timeSignatureFraction 72
tiny 210
tiny 233
tocItem 750
top-margin 505
top-markup-spacing 507
top-system-spacing 507
translate 235
translate-scaled 235
transpose 5, 10, 13, 750
transposedCueDuring 208, 750
transposition 24, 202, 750
treCorde 317
tremolo 157
tremoloFlags 158
triangle 239
trill 140
tuplet 45
tuplet 72, 750
tupletDown 45
tupletNeutral 45
TupletNumber 46
tupletNumberFormatFunction 46
tupletSpan 750
tupletSpannerDuration 46
tupletUp 45
tweak 750
two-sided 508
type 568

U

unaCorda 317
underline 231
undo 751
unfold 153
unfoldRepeats 751
unHideNotes 214

V

VaticanaStaff 181
VerticalAxisGroup 523
voice 26, 28
Voice 164
voiceOne 164
void 751

W

walkerHeads 38
walkerHeadsMinor 39
whichBar 99
with-color 215
withMusicProperty 751
wordwrap 236
wordwrap-lines 242

X

X-offset 523
x11-color 215, 216
xNote 751
xNotesOn 751

Appendice E Indice di LilyPond

Oltre a tutti i comandi e le parole chiave di LilyPond, questo indice elenca i termini musicali e le espressioni che si riferiscono a ognuno di essi, corredati di collegamenti alle relative sezioni del manuale. Ogni collegamento è composto da due parti. La prima parte porta al punto esatto del manuale in cui compare l'argomento; la seconda parte porta all'inizio della sezione del manuale in cui l'argomento è trattato.

!

! 6

"

"|" 104

'

' 1

,

, ... 1

-

- 115

.

. 43

/

/ 397
/+ 397

:

: 158

<

< 159
<...> 159

=

= 9

>

> 159

?

? 6

[

[..... 89

]

] 89

^

^ 396

_

_ 255

\

\! 118
\(..... 129
\) 129
\< 118
\> 118
\abs-fontsize 232, 653
\absolute 2
\accent 115
\accepts 568, 569, 570
\acciaccatura 107
\accidentalStyle 26
\addChordShape 355
\addInstrumentDefinition 201, 209
\addlyrics 249, 251, 252
\addQuote 202
\aeolian 21
\afterGrace 108
\aikenHeads 38
\aikenHeadsMinor 39
\alias 568
\allowPageTurn 519
\alterBroken 609
\alternative 143
\appendToTag 483
\appoggiatura 107
\arpeggio 137
\arpeggioArrowDown 137
\arpeggioArrowUp 137
\arpeggioBracket 137
\arpeggioNormal 137
\arpeggioParenthesis 137
\arpeggioParenthesisDashed 137
\arrow-head 239
\arrow-head 677
\ascendens 429, 435
\auctum 429, 435
\augmentum 435
\auto-footnote 697
\autoBeamOff 78, 315
\autoBeamOn 78
\autochange 313

\backslashed-digit ... 697
\balloonGrobText ... 218
\balloonLengthOff ... 218
\balloonLengthOn ... 218
\balloonText ... 218
\bar ... 93, 99
\barNumberCheck ... 105
\beam ... 677
\bendAfter ... 132
\bold ... 231, 654
\book ... 451, 454
\bookOutputName ... 453
\bookOutputSuffix ... 453
\bookpart ... 452, 454, 517
\box ... 237, 654
\bracket ... 124, 237, 677
\break ... 517
\breathe ... 130
\breve ... 43, 54
\cadenzaOff ... 70
\cadenzaOn ... 70
\caesura ... 427
\caps ... 654
\cavum ... 429, 435
\center-align ... 234, 662
\center-column ... 236, 662
\change ... 312
\char ... 698
\chordmode ... 5, 13, 353
\chordRepeats ... 328
\chords ... 399
\circle ... 237, 677
\clef ... 17
\cm ... 586
\coda ... 115
\column ... 236, 663
\column-lines ... 703
\combine ... 239
\combine ... 663
\compoundMeter ... 74
\compressFullBarRests ... 58, 59
\concat ... 663
\consists ... 568
\context ... 555, 563
\context in \layout block ... 563
\cr ... 118
\cresc ... 119
\crescHairpin ... 119
\crescTextCresc ... 119
\crossStaff ... 315
\cueClef ... 205
\cueDuring ... 205
\cueDuringWithClef ... 205
\customTabClef ... 684
\decr ... 118
\decresc ... 119
\defaultchild ... 571
\defaultTimeSignature ... 62
\defineBarLine ... 97
\deminutum ... 429, 435
\denies ... 568, 569, 570
\descendens ... 429, 435
\dim ... 119
\dimHairpin ... 119
\dimTextDecr ... 119
\dimTextDecresc ... 119
\dimTextDim ... 119
\dir-column ... 664
\discant ... 693
\displayLilyMusic ... 500
\divisioMaior ... 427
\divisioMaxima ... 427
\divisioMinima ... 427
\dorian ... 21
\dotsDown ... 44
\dotsNeutral ... 44
\dotsUp ... 44
\doubleflat ... 684
\doublesharp ... 684
\downbow ... 115, 321
\downmordent ... 115
\downprall ... 115
\draw-circle ... 239
\draw-circle ... 678
\draw-dashed-line ... 678
\draw-dotted-line ... 678
\draw-hline ... 679
\draw-line ... 239
\draw-line ... 679
\drummode ... 181
\dynamic ... 124, 654
\dynamicDown ... 120
\dynamicNeutral ... 120
\dynamicUp ... 120
\easyHeadsOff ... 37
\easyHeadsOn ... 37
\ellipse ... 679
\epsfile ... 239
\epsfile ... 680
\espressivo ... 115, 119
\expandFullBarRests ... 58, 59
\eyeglasses ... 698
\f ... 118
\featherDurations ... 92
\fermata ... 115, 684
\fermataMarkup ... 58, 59, 115
\ff ... 118
\fff ... 118
\ffff ... 118
\fffff ... 118
\fill-line ... 236, 664
\fill-with-pattern ... 665
\filled-box ... 239
\filled-box ... 680
\finalis ... 427
\finger ... 212, 655
\flageolet ... 115
\flat ... 685
\flexa ... 435
\fontCaps ... 655
\fontsize ... 232, 655
\footnote ... 468, 698
\fp ... 118
\fraction ... 698
\freeBass ... 693
\frenchChords ... 402
\fret-diagram ... 343, 689
\fret-diagram-terse ... 345, 690
\fret-diagram-verbose ... 347, 690
\fromproperty ... 698

\funkHeads ... 38
\funkHeadsMinor ... 39
\general-align ... 235, 665
\germanChords ... 402
\glissando ... 133
\grace ... 107
\halfopen ... 115
\halign ... 234, 666
\harmonic ... 322, 330
\harmonicByFret ... 330
\harmonicByRatio ... 330
\harmonicsOff ... 322
\harmonicsOn ... 322
\harp-pedal ... 691
\hbracket ... 237, 680
\hcenter-in ... 667
\header ... 454
\hide ... 592
\hideKeySignature ... 382
\hideNotes ... 214
\hideSplitTiedTabNotes ... 329
\hideStaffSwitch ... 315
\hspace ... 668
\huge ... 210
\huge ... 233
\huge ... 655
\improvisationOff ... 41, 76
\improvisationOn ... 41, 76
\in ... 586
\inclinatum ... 429, 435
\include ... 480
\inStaffSegno ... 146
\instrumentSwitch ... 201
\inversion ... 13
\ionian ... 21
\italianChords ... 402
\italic ... 231, 655
\justified-lines ... 242, 703
\justify ... 236, 669
\justify-field ... 668
\justify-string ... 669
\keepWithTag ... 483
\key ... 20, 39
\killCues ... 209
\label ... 478
\laissezVibrer ... 51
\large ... 210
\large ... 233
\large ... 656
\larger ... 232
\larger ... 233
\larger ... 656
\layout ... 454, 512
\layout ... 553
\layout ... 563
\left-align ... 234, 670
\left-brace ... 699
\left-column ... 670
\lheel ... 115
\line ... 670
\linea ... 429, 435
\lineprall ... 115
\locrian ... 21
\longa ... 43, 54
\longfermata ... 115
\lookup ... 699
\lower ... 234, 671
\ltoe ... 115
\lydian ... 21
\lyricmode ... 248, 249
\lyricsto ... 249, 251
\magnify ... 232, 656
\major ... 21
\makeClusters ... 164
\map-markup-commands ... 704
\marcato ... 115
\mark ... 105, 226
\markalphabet ... 699
\markLengthOff ... 67, 227
\markLengthOn ... 67, 227
\markletter ... 700
\markup ... 226, 228, 229, 230
\markuplist ... 229, 242, 243
\maxima ... 43, 54
\medium ... 656
\melisma ... 255
\melismaEnd ... 255
\mergeDifferentlyDottedOff ... 168
\mergeDifferentlyDottedOn ... 168
\mergeDifferentlyHeadedOff ... 168
\mergeDifferentlyHeadedOn ... 168
\mf ... 118
\midi ... 454
\midi ... 553
\minor ... 21
\mixolydian ... 21
\mm ... 586
\modalInversion ... 15
\modalTranspose ... 14
\mordent ... 115
\mp ... 118
\musicglyph ... 106, 685
\name ... 568
\natural ... 685
\new ... 555
\newSpacingSection ... 540
\noBeam ... 90
\noBreak ... 517
\noPageBreak ... 518
\noPageTurn ... 519
\normal-size-sub ... 657
\normal-size-super ... 232, 657
\normal-text ... 657
\normalsize ... 210
\normalsize ... 233
\normalsize ... 658
\note ... 686
\note-by-number ... 685
\null ... 234, 700
\number ... 658
\numericTimeSignature ... 62
\octaveCheck ... 9
\omit ... 592
\on-the-fly ... 467, 700
\once ... 577
\once ... 578
\oneVoice ... 164
\open ... 115, 321
\oriscus ... 429, 435
\ottava ... 23

\oval ... 680
\override ... 577, 581, 700
\override-lines ... 704
\overrideProperty ... 581
\overrideTimeSignatureSettings ... 62
\p ... 118
\pad-around ... 238, 671
\pad-markup ... 238, 671
\pad-to-box ... 238, 672
\pad-x ... 238, 672
\page-link ... 701
\page-ref ... 478, 701
\pageBreak ... 518
\pageTurn ... 519
\paper ... 454, 503
\parallelMusic ... 178
\parenthesize ... 216, 681
\partcombine ... 173, 276
\partcombine and lyrics ... 276
\partcombine e testo vocale ... 175
\partcombineApart ... 174
\partcombineAutomatic ... 174
\partcombineChords ... 174
\partcombineSoloI ... 174
\partcombineSoloII ... 174
\partcombineUnisono ... 174
\partial ... 69, 143, 144
\path ... 681
\pattern ... 701
\pes ... 435
\phrasingSlurDashed ... 129
\phrasingSlurDashPattern ... 129
\phrasingSlurDotted ... 129
\phrasingSlurDown ... 129
\phrasingSlurHalfDashed ... 129
\phrasingSlurHalfSolid ... 129
\phrasingSlurNeutral ... 129
\phrasingSlurSolid ... 129
\phrasingSlurUp ... 129
\phrygian ... 21
\pitchedTrill ... 141
\portato ... 115
\postscript ... 239
\postscript ... 682
\powerChords ... 368
\pp ... 118
\ppp ... 118
\pppp ... 118
\ppppp ... 118
\prall ... 115
\pralldown ... 115
\prallmordent ... 115
\prallprall ... 115
\prallup ... 115
\predefinedFretboardsOff ... 362
\predefinedFretboardsOn ... 362
\property-recursive ... 701
\pt ... 586
\pushToTag ... 483
\put-adjacent ... 672
\quilisma ... 429, 435
\quoteDuring ... 202, 205
\raise ... 234, 672
\relative ... 2, 5, 13, 314
\RemoveEmptyStaves ... 195, 197
\removeWithTag ... 483
\repeat ... 143
\repeat percent ... 155
\repeat tremolo ... 157
\repeatTie ... 51, 146, 272
\replace ... 658
\rest ... 54, 687
\rest-by-number ... 686
\retrograde ... 14
\reverseturn ... 115
\revert ... 578
\revertTimeSignatureSettings ... 64
\rfz ... 118
\rheel ... 115
\right-align ... 234, 673
\right-brace ... 701
\right-column ... 673
\rightHandFinger ... 364
\roman ... 658
\rotate ... 673
\rounded-box ... 237, 683
\rtoe ... 115
\sacredHarpHeads ... 38
\sacredHarpHeadsMinor ... 39
\sans ... 659
\scale ... 683
\scaleDurations ... 50, 72
\score ... 450, 454, 687
\segno ... 115
\semiflat ... 688
\semiGermanChords ... 402
\semisharp ... 688
\sesquiflat ... 688
\sesquisharp ... 689
\set ... 81, 575, 581
\sf ... 118
\sff ... 118
\sfz ... 118
\shape ... 606
\sharp ... 689
\shiftOff ... 168
\shiftOn ... 168
\shiftOnn ... 168
\shiftOnnn ... 168
\shortfermata ... 115
\showKeySignature ... 382
\showStaffSwitch ... 315
\signumcongruentiae ... 115
\simple ... 659
\skip ... 56, 271
\slashed-digit ... 701
\slashedGrace ... 107
\slurDashed ... 126
\slurDashPattern ... 127
\slurDotted ... 126
\slurDown ... 126
\slurHalfDashed ... 127
\slurHalfSolid ... 127
\slurNeutral ... 126
\slurSolid ... 126
\slurUp ... 127
\small ... 210
\small ... 233
\small ... 659
\smallCaps ... 659

\smaller ... 232
\smaller ... 233
\smaller ... 660
\snappizzicato ... 115
\sostenutoOff ... 317
\sostenutoOn ... 317
\southernHarmonyHeads ... 38
\southernHarmonyHeadsMinor ... 39
\sp ... 118
\spp ... 118
\staccatissimo ... 115
\staccato ... 115
\startGroup ... 221
\startStaff ... 188, 192
\startTrillSpan ... 140
\stdBass ... 694
\stdBassIV ... 695
\stdBassV ... 695
\stdBassVI ... 696
\stemDown ... 217
\stemNeutral ... 217
\stemUp ... 217
\stencil ... 702
\stopGroup ... 221
\stopped ... 115
\stopStaff ... 188, 192, 195
\stopTrillSpan ... 140
\storePredefinedDiagram ... 355
\stringTuning ... 340
\stropha ... 429, 435
\strut ... 702
\sub ... 232, 660
\super ... 232, 660
\sustainOff ... 317
\sustainOn ... 317
\tabChordRepeats ... 328
\tabFullNotation ... 327
\table-of-contents ... 480, 704
\tag ... 483
\taor ... 382
\teeny ... 210
\teeny ... 233
\teeny ... 661
\tempo ... 66
\tenuto ... 115
\text ... 661
\textLengthOff ... 59, 223
\textLengthOn ... 59, 223
\textSpannerDown ... 224
\textSpannerNeutral ... 224
\textSpannerUp ... 224
\thumb ... 115, 212
\tied-lyric ... 689
\tieDashed ... 52
\tieDotted ... 52
\tieDown ... 52
\tieNeutral ... 52
\tieSolid ... 52
\tieUp ... 52
\time ... 61, 81
\tiny ... 210
\tiny ... 233
\tiny ... 661
\tocItem ... 480
\translate ... 235, 674
\translate-scaled ... 235, 674
\transparent ... 702
\transpose ... 5, 10, 13
\transposedCueDuring ... 208
\transposition ... 24, 202
\treCorde ... 317
\triangle ... 239
\triangle ... 684
\trill ... 115, 140
\tuplet ... 45
\tuplet ... 72
\tupletDown ... 45
\tupletNeutral ... 45
\tupletUp ... 45
\turn ... 115
\tweak ... 579, 581
\type ... 568
\typewriter ... 661
\unaCorda ... 317
\underline ... 231, 661
\unfoldRepeats ... 495
\unHideNotes ... 214
\unset ... 576
\upbow ... 115, 321
\upmordent ... 115
\upprall ... 115
\upright ... 662
\varcoda ... 115
\vcenter ... 674
\verbatim-file ... 702
\verylongfermata ... 115
\virga ... 429, 435
\virgula ... 427
\voiceFourStyle ... 168
\voiceNeutralStyle ... 168
\voiceOne ... 164
\voiceOne ... \voiceFour ... 164
\voiceOneStyle ... 168
\voiceThreeStyle ... 168
\voiceTwoStyle ... 168
\void ... 501
\vspace ... 674
\walkerHeads ... 38
\walkerHeadsMinor ... 39
\whiteout ... 702
\with ... 561, 566
\with-color ... 215, 703
\with-dimensions ... 703
\with-link ... 703
\with-url ... 684
\woodwind-diagram ... 692
\wordwrap ... 236, 675
\wordwrap-field ... 675
\wordwrap-internal ... 704
\wordwrap-lines ... 242, 704
\wordwrap-string ... 676
\wordwrap-string-internal ... 704

|

| ... 104

~

~ 50

1

15ma 23

8

8va 23
8ve 23

A

a capo, testo 236
abbellimenti 107, 115
abbellimenti al termine di una nota 108
abbellimenti all'interno della parentesi di un gruppo irregolare 49
abbellimenti, modifica delle impostazioni di formattazione 109
abbellimenti, modifica manuale 109
`absolute` 2, 742
accelerando in MIDI 494
accent 706
accento 115
accentus 706
`accepts` 568
acciaccatura 107
`acciaccatura` 742
acciaccatura su più note 111
Accidental, musica ficta 422
accidentals 422, 426, 437
`accidentalStyle` 742
accollatura 182
accordi 159
accordi e legature di valore 51
accordi e ottava relativa 4
accordi per chitarra, tabella 76
accordi vuoti 160
accordi, alterazioni in 32
accordi, altezza relativa 160
accordi, diteggiatura 212
accordion 318
accordion discant symbols 318
accordion shift symbols 318
accordion shifts 318
accordo, ripetizione 161
`addChordShape` 355, 742
adding a white background to text 702
adding custom fret diagrams 354
`addInstrumentDefinition` 201, 209, 742
`additionalPitchPrefix` 401
additions, in chords 395
`addQuote` 202, 742
adjusting staff symbol 586
`aeolian` 21
`afterGrace` 108, 743
afterGraceFraction 711
Aiken, testa di nota 38
`aikenHeads` 38
`aikenHeadsMinor` 39
aiuto, nuvoletta 218
al niente 121
`alias` 568
align to objects 602
`alignAboveContext` 571
`alignBelowContext` 270, 571
alist 709
allineamento orizzontale del testo 234
allineamento sulla cadenza 113
allineamento verticale del testo 234
allineamento, testo, comandi 237
allineare il markup 234
allineare il testo 234
`allowPageTurn` 743
`allowVoltaHook` 743
alterazione 5
alterazione di cortesia 6
alterazione di sicurezza 6
alterazione di un quarto di tono 7
alterazione e legatura di valore 6
alterazione tra parentesi 6
alterazione, di cortesia 6
alterazione, di sicurezza 6
alterazione, quarto di tono 7
alterazione, tra parentesi 6
alterazioni 26
alterazioni automatiche 26
alterazioni di precauzione in stile moderno 29
alterazioni e note simultanee 32
alterazioni in stile moderno 29
alterazioni moderne 29
alterazioni negli accordi 32
alterazioni su più voci 29
alterazioni, cadenze 70
alterazioni, musica in tempo libero 70
alterazioni, stile *modern-cautionary* 29
alterazioni, stile moderno delle 28
`alterBroken` 743
altered chords 395
alternative endings and lyrics 271
alternative melody, switching to 281
altezza naturale 5
altezza relativa, accordi 160
altezze 1
altezze, trasposizione delle 10
alto, chiave di 17
Amazing Grace bagpipe example 382
ambito delle altezze 33
ambitus 33
anacrusi 69
anacrusi in una ripetizione 144
analisi musicologica 221
angled hairpins 599
annidamento dei righi 186
annidamento, ripetizioni 150
`annotate-spacing` 549
annotazione 230
annotazione su pausa multipla 58
anthems 286
antica, chiave 17
aperto (\open) 115
apice 232
`appendToTag` 743
`applyContext` 743
`applyMusic` 743
`applyOutput` 743
appoggiatura 107

appoggiatura ... 743
Arabic key signatures ... 445
Arabic music ... 444
Arabic music example ... 447
Arabic music template ... 447
Arabic note names ... 444
Arabic semi-flat symbol ... 445
Arabic time signatures ... 447
arcata in giù ... 115
arcata in su ... 115
armatura di chiave ... 5, 20
armonico (\flageolet) ... 115
armonico, testa di nota ... 35
arpeggio ... 137
arpeggio attraverso il rigo, stile della parentesi ... 140
arpeggio spezzato ... 137
arpeggio, simboli speciali ... 137
arpeggioArrowDown ... 137
arpeggioArrowUp ... 137
arpeggioBracket ... 137
arpeggioNormal ... 137
arpeggioParenthesis ... 137
arpeggioParenthesisDashed ... 137
arrow-head ... 239
Articlulate scripts ... 494
articolazione "espressivo" ... 119
articolazioni ... 115
articulation-event ... 204
articulations ... 427
articulations in MIDI ... 494
artificial harmonics ... 322
assertBeamQuant ... 743
assertBeamSlope ... 743
associatedVoice ... 249
associatedVoice ... 251, 281
association list ... 709
assoluta, ottava ... 1
assoluto ... 1
aug ... 394
auto-first-page-number ... 511
autoBeaming ... 81
autoBeaming ... 553
autoBeamOff ... 78
autoBeamOn ... 78
autochange ... 313, 743
autochange and relative music ... 314
automatic chord diagrams ... 361
automatic fret diagrams ... 361
automatic staff changes ... 313
automaticBars ... 597

B

backslashed digits ... 697
bagpipe ... 382
bagpipe example ... 382
Balloon_engraver ... 218
balloonGrobText ... 218, 743
balloonLengthOff ... 218
balloonLengthOn ... 218
balloonText ... 218, 743
banjo tablature ... 324
banjo tablatures ... 369
banjo tunings ... 370
banjo-c-tuning ... 370
banjo-modal-tuning ... 370
banjo-open-d-tuning ... 370
banjo-open-dm-tuning ... 370
bar ... 93, 99, 743
bar lines, suppressing ... 597
barCheckSynchronize ... 104
baritono, chiave di ... 17
BarNumber ... 100
barNumberCheck ... 105, 743
barNumberVisibility ... 100
barrata, testa di nota ... 35
barre indications ... 343
Bartók pizzicato ... 323
bartype ... 99
base-shortest-duration ... 539
baseMoment ... 81
bass note, for chords ... 397
Bass, figured ... 406
Bass, thorough ... 406
Basso continuo ... 406
basso, chiave di ... 17
battiti per minuto ... 66
battuta in levare ... 69
battuta, controlli ... 104
battuta, numeri ... 100
battuta, numero, formato del ... 101
battuta, stanghette ... 93
battuta, stanghette manuali ... 94
battute dei ritornelli ... 93
beamExceptions ... 81
beams, cross-staff ... 312
beatStructure ... 81
bemolle ... 5
bemolle, doppio ... 5
bendAfter ... 132, 743
Bézier curves, control points ... 605
binari ferroviari ... 131
binding gutter ... 508
binding-offset ... 508
bisbiglando ... 319
Bison ... 710
blank-after-score-page-penalty ... 510
blank-last-page-penalty ... 510
blank-page-penalty ... 510
BNF ... 710
bold ... 231
bookOutputName ... 743
bookOutputSuffix ... 743
bookTitleMarkup ... 464
bottom-margin ... 505
bowing indications ... 321
box ... 237
bracket ... 124, 237, 317
break-align-symbols ... 602
break-visibility ... 593
breakable ... 79
breakbefore ... 461
breaking lines ... 515
breaking pages ... 542
breathe ... 130, 744
breve ... 43, 54
broken spanners, modifying ... 609

C

cadenza ... 70, 113
cadenza, allineamento su ... 113
cadenza, alterazioni ... 70
cadenza, interruzioni di linea ... 72
cadenza, interruzioni di pagina ... 72
cadenza, numeri di battuta ... 70
cadenza, stanghette ... 70
cadenza, travature ... 70
`cadenzaOff` ... 70
`cadenzaOn` ... 70
callback ... 709
cambiare i tipi di carattere ... 231
cambio di strumento ... 201
capo ... 347
caratteri riservati, stampare ... 230
caratteri speciali in modalità markup ... 230
`center-align` ... 234
`center-column` ... 236
centered dynamics in piano music ... 311
centering a column of text ... 662
centrare il testo sulla pagina ... 236
cesura ... 131
`change` ... 312
changing direction of text columns ... 664
changing properties ... 575
changing staff automatically ... 313
changing staff manually ... 312
chants ... 297
character names ... 290
`check-consistency` ... 508
chiave ... 5, 17
chiave antica ... 17
chiave di baritono ... 17
chiave di basso ... 17
chiave di contralto ... 17
chiave di Do ... 17
chiave di Fa ... 17
chiave di mezzosoprano ... 17
chiave di Sol ... 17
chiave di soprano ... 17
chiave di subbasso ... 17
chiave di tenore ... 17
chiave di tenore per coro ... 18
chiave di varbaritono ... 17
chiave di violino ... 17
chiave francese ... 17
chiave traspositrice ... 18
chitarra, teste di nota ... 35
chiuso (\stopped) ... 115
chord chords ... 392
chord diagrams ... 342, 351
chord diagrams, automatic ... 361
chord glissandi ... 337
chord inversions ... 397
chord mode ... 392
chord names ... 392, 398
Chord names in MIDI ... 494
chord names with fret diagrams ... 353
chord quality ... 393
chord shapes for fretted instruments ... 355
chord steps, altering ... 396
chord, modifying one note in ... 579
Chord, repetition ... 328
`chordChanges` ... 399
`chordmode` ... 5, 13, 353
`chordNameExceptions` ... 402
`chordNameLowercaseMinor` ... 400
`ChordNames` ... 353
`chordNameSeparator` ... 401
`chordNoteNamer` ... 401
`chordPrefixSpacer` ... 402
`chordRepeats` ... 744
`chordRootNamer` ... 401
chords ... 398
chords, cross-staff ... 315
chords, jazz ... 400
chords, power ... 368
chords, splitting across staves with \autochange ... 314
chords, suppressing repeated ... 399
Christian Harmony, testa di nota ... 38
`circle` ... 237
circling text ... 677
circulus ... 706
citare le voci ... 202
citazioni in corpo più piccolo, togliere le ... 209
`clef` ... 17, 744
clef, moderntab ... 341
clef, percussion ... 371
clef, tab ... 341
clef, visibility following explicit change ... 595
clefs ... 417, 425, 436
clefs, visibility of transposition ... 597
closure ... 709
cluster ... 164
cluster di note ... 164
coda ... 106, 115, 706
coda sulla stanghetta ... 226
collisione, numeri di battuta ... 104
collisioni ... 168
collisioni di note ... 168
collisioni, ignorare ... 163
collisioni, troppe colonne di note che si urtano ... 163
collisioni,ignorare ... 173
collisions, cross-staff voices ... 312
colonne, testo ... 236
`color` ... 215
colorare gli oggetti ... 215
colorare le note ... 215
colorare le voci ... 168
colorate, note ... 215
colorati, oggetti ... 215
colore negli accordi ... 216
colore rgb ... 215
colore x11 ... 216
colori ... 215
coloring objects ... 593
coloring text ... 703
Colors, list of ... 631
`column` ... 236
combinatore delle parti ... 173
combinazione automatica delle parti ... 173
`combine` ... 239
comma intervals ... 449
`common-shortest-duration` ... 539
`Completion_heads_engraver` ... 75
`Completion_rest_engraver` ... 75
`compoundMeter` ... 744

`compressFullBarRests` ... 58, 59
comprimere la musica ... 50
concatenating text ... 663
condensare le pause normali ... 61
`consists` ... 568
constante-hairpins ... 121
context properties, changing defaults ... 563
Contexts, creating and referencing ... 555
contexts, defining new ... 568
contexts, implicit ... 571
contexts, keeping alive ... 558
contexts, layout order ... 570
contexts, lifetime ... 558
control points, Bézier curves ... 605
control points, tweaking ... 581
controlli del numero di battuta ... 104
controlli di battuta ... 104
controlling general text alignment ... 665
controllo dell'ottava ... 9
controllo della misura ... 104
controllo delle altezze ... 9
`controlpitch` ... 9
copyright sign ... 488
coro, rigo per ... 182
corona ... 106, 115
corona sulla stanghetta ... 226
correzione dell'ottava ... 9
`cr` ... 118
creating empty text objects ... 700
creating horizontal spaces in text ... 668
creating text fractions ... 698
creating vertical spaces in text ... 674, 702
creazione del rigo ... 181
`cresc` ... 119
crescendo ... 118
crescendo-event ... 204
`crescHairpin` ... 119
`crescTextCresc` ... 119
`cross` ... 35
cross staff chords ... 315
cross staff line ... 315
cross staff notes ... 315
cross staff stems ... 315
cross-staff ... 315
cross-staff beams ... 312
cross-staff chords ... 315
cross-staff collisions ... 312
cross-staff line ... 315
cross-staff notes ... 312, 315
cross-staff stems ... 315
`crossStaff` ... 744
`cueClef` ... 205, 744
`cueClefUnset` ... 744
`cueDuring` ... 205, 744
`cueDuringWithClef` ... 205, 744
cues, musical ... 292
CueVoice ... 205
`currentBarNumber` ... 100, 114
custodes ... 416
custom fret diagrams ... 342
custom fret diagrams, adding ... 354
custom string tunings ... 340
customized fret diagram ... 348
customizing chord names ... 400
custos ... 416

D

D.S. al Fine ... 106
dampened notes on fretted instruments ... 366
`deadNote` ... 744
decorazione del testo ... 237
`decr` ... 118
`decresc` ... 119
decrescendo ... 118
`default` ... 26, 28
default context properties, changing ... 563
`default-staff-staff-spacing` ... 523
`defaultBarType` ... 99
`defaultNoteHeads` ... 744
`defaultTimeSignature` ... 62
`defineBarLine` ... 97, 744
definire le stanghette ... 97
delimitatori di inizio del sistema ... 182
delimitatori di inizio del sistema annidati ... 186
`denies` ... 568
diagram, fret, customized ... 348
diagramma degli accordi per chitarra ... 76
diagrams, chord for fretted instruments ... 342
diagrams, fret ... 342
diagrams, fret, transposing ... 353
diamante, testa di nota ... 35
diamond-shaped note heads ... 322
diesis ... 5
diesis, doppio ... 5
`dim` ... 119
`dim` ... 394
dimensione del testo ... 232
dimensione del tipo di carattere ... 232
dimensione del tipo di carattere (elementi della notazione) ... 210
dimensione del tipo di carattere standard (per gli elementi della notazione) ... 211
`dimHairpin` ... 119
diminuendo ... 118
`dimTextDecr` ... 119
`dimTextDecresc` ... 119
`dimTextDim` ... 119
dinamica, nuovi segni di ... 124
dinamiche ... 118
dinamiche assolute ... 118
dinamiche editoriali ... 124
dinamiche, parentesi ... 124
dinamiche, posizionamento verticale ... 120
discant symbols, accordion ... 318
disegnare oggetti grafici ... 237
`displayLilyMusic` ... 744
`displayMusic` ... 744
`displayScheme` ... 744
disposizione delle travature, proprietà predefinite delle indicazioni di tempo ... 62
distance between staves ... 523
distances, absolute ... 586
distances, scaled ... 586
dita, cambio ... 212
diteggiatura ... 212
diteggiatura per accordi ... 212
diteggiature e pause multiple ... 61
divided lyrics ... 275
divisio ... 426
divisione delle note ... 75

divisione delle pause ... 75
divisiones ... 426
Do, chiave di ... 17
dodecaphonic ... 31
dodecaphonic, stile delle alterazioni ... 31
doppio bemolle ... 5
doppio diesis ... 5
doppio mordente (\prallmordent) ... 115
doppio mordente (\prallprall) ... 115
doppio punto, note ... 43
dorian ... 21
dorico ... 21
dotsDown ... 44
dotsNeutral ... 44
dotsUp ... 44
down bow indication ... 321
downbow ... 706
downmordent ... 706
downprall ... 706
draw-circle ... 239
draw-line ... 239
drawing a line across a page ... 679
drawing beams within text ... 677
drawing boxes with rounded corners ... 680
drawing boxes with rounded corners around text ... 683
drawing circles within text ... 678
drawing dashed lines within text ... 678
drawing dotted lines within text ... 678
drawing ellipse around text ... 679
drawing lines within text ... 679
drawing oval around text ... 680
drawing paths ... 681
drawing solid boxes within text ... 680
drawing staff symbol ... 586
drawing triangles within text ... 684
drummode ... 181
drums ... 370, 372
DrumStaff ... 181
durata delle note ... 43
durata predefinita ... 43
durate, scalare ... 49
dynamic ... 124
dynamic-event ... 204
dynamicDown ... 120
DynamicLineSpanner ... 120
dynamicNeutral ... 120
dynamics, centered in keyboard music ... 311
dynamicUp ... 120

E

easyHeadsOff ... 37
easyHeadsOn ... 37
editoriali, dinamiche ... 124
elementi testuali non vuoti ... 223
elencare i tipi di carattere disponibili ... 245
encapsulated postscript output ... 490
enclosing text in a box with rounded corners ... 683
enclosing text within a box ... 654
endSpanners ... 744
engravers, including in contexts ... 568
entering lyrics ... 248
eolio ... 21
EPS output ... 490
epsfile ... 239
espandere la musica ... 50
espressioni di markup ... 230
espressivo ... 115
espressivo ... 119
espressivo ... 706
estensione ... 33
estensori del testo ... 224
estensori del testo, formattazione ... 224
eventChords ... 745
eventi segnaposto ... 160
exceptions, chord names ... 403
expandFullBarRests ... 58, 59
explicitClefVisibility ... 595
explicitKeySignatureVisibility ... 595
extended chords ... 395
extender ... 259
extra-offset ... 523

F

f ... 118
Fa, chiave di ... 17
famiglie di tipi di carattere ... 233
famiglie di tipi di carattere, impostare ... 245
featherDurations ... 92, 745
fermata ... 706
fermata su pausa multipla ... 58
fermataMarkup ... 58, 59
Ferneyhough, forcelle ... 121
Feta font ... 632
ff ... 118
fff ... 118
ffff ... 118
fffff ... 118
Figured bass ... 406
figured bass alignment ... 411
figured bass extender lines ... 409
fill-line ... 236
filled-box ... 239
finali alternati, ripetizioni ... 153
finali alternativi ... 143
finali alternativi con legature di valore ... 146
finalis ... 426
fine ripetizione ... 151
finger ... 212, 745
fingering vs. string numbers ... 324
fingerings, adding to fret diagrams ... 363
fingerings, right hand for fretted instruments ... 364
first-page-number ... 511
flageolet ... 706
flags ... 420
flared-hairpin ... 121
Flex ... 710
follow voice ... 315
followVoice ... 315
font ... 243, 709
font non testuali nel markup ... 243
font size, setting ... 514
font, cambiare ... 231
Font, Feta ... 632
font-interface ... 211, 243
font-size ... 210, 211
fontsize ... 232

`fontSize` ... 210
`footnote` ... 745
footnotes ... 468
footnotes in music expressions ... 468
footnotes in stand-alone text ... 474
footnotes, event-based ... 469
footnotes, time-based ... 471
forcella ... 118
forcelle allargate (flared-hairpins) ... 121
forcelle continue (constante-hairpins) ... 121
forcelle Ferneyhough ... 121
`forget` ... 31
forget, stile delle alterazioni ... 31
formato del segno di chiamata ... 106
formattare gli estensori del testo ... 224
formattare le notine ... 205
formattazione del gruppo irregolare ... 46
formattazione della terzina ... 46
formattazione mensurale ... 185
formatting in lyrics ... 248
four bar music ... 516
`four-string-banjo` ... 370
`fp` ... 118
frammenti ... 205
frammenti, citare i ... 202
francese, chiave ... 17
frase, legature di ... 129
fret ... 327
fret diagram, customized ... 348
fret diagrams ... 342, 351
fret diagrams with chord names ... 353
fret diagrams, adding custom ... 354
fret diagrams, adding fingerings ... 363
fret diagrams, automatic ... 361
fret diagrams, custom ... 342
fret diagrams, mandolin ... 351
fret diagrams, transposing ... 353
fret diagrams, ukulele ... 351
`fret-diagram` ... 343
fret-diagram markup ... 343
`fret-diagram-interface` ... 348
`fret-diagram-terse` ... 345
fret-diagram-terse markup ... 345
`fret-diagram-verbose` ... 347
fret-diagram-verbose markup ... 347
`FretBoards` ... 351
fretted instruments, chord shapes ... 355
fretted instruments, dampened notes ... 366
fretted instruments, harmonics ... 366
fretted instruments, indicating position and barring ... 366
fretted instruments, predefined string tunings ... 339
fretted instruments, right hand fingerings ... 364
frigio ... 21
Funk, testa di nota ... 38
`funkHeads` ... 38
`funkHeadsMinor` ... 39

G

gambo ... 217
gambo barrato ... 109
gambo invisibile ... 217
gambo, direzione ... 217
gambo, giù ... 217
gambo, neutrale ... 217
gambo, su ... 217
`general-align` ... 235
gestione del tempo ... 114
ghost notes ... 216
giustificato, testo ... 236
glifi musicali ... 106
glissandi e ripetizioni ... 150
`glissando` ... 133
global variable ... 711
glyph ... 709
`grace` ... 745
grace notes ... 382
grace notes and lyrics ... 280
graffa verticale ... 182
graffe, varie dimensioni ... 243
grafica inclusa ... 239
grafica, inclusione ... 237
grammar for LilyPond ... 710
graphical object interfaces ... 710
graphical objects ... 709
Gregorian square neumes ligatures ... 428
`GregorianTranscriptionStaff` ... 181
`Grid_line_span_engraver` ... 219
`Grid_point_engraver` ... 219
`gridInterval` ... 219
griglie ... 219
grob ... 572, 709
grob properties ... 577
grob-interface ... 710
`grobdescriptions` ... 745
grobs, overwriting ... 593
grobs, visibility of ... 592
`grow-direction` ... 92
gruppetto (\turn) ... 115
gruppetto rovesciato (\reverseturn) ... 115
gruppi irregolari ... 45
gruppi irregolari, raggruppamento ... 45
gruppo di righi ... 182
gruppo irregolare, formattazione del ... 46
gruppo irregolare, modifiche del numero del ... 46
gruppo irregolare, posizionamento della parentesi quadra ... 45
guitar tablature ... 324
gutter ... 508

H

hairpins, angled ... 599
halfopen ... 706
`halign` ... 234
hammer on ... 338
harmonic indications in tablature notation ... 330
Harmonica Sacra, testa di nota ... 38
`harmonicByFret` ... 745
`harmonicByRatio` ... 745
`harmonicNote` ... 745
harmonics on fretted instruments ... 366
harmonics, artificial ... 322
harmonics, natural ... 322
`harmonicsOn` ... 745
harp pedal diagrams ... 319
harp pedals ... 319
harps ... 319

`hbracket` ... 237
`hide` ... 745
`hideKeySignature` ... 382
`hideNotes` ... 214
`hideStaffSwitch` ... 315
horizontal spacing ... 538
horizontal spacing, overriding ... 611
`horizontal-shift` ... 509
`Horizontal_bracket_engraver` ... 221
horizontally centering text ... 662
hufnagel ... 414
`huge` ... 210
`huge` ... 233
hymns ... 297
hyphens ... 259

I

ictus ... 706
immagini incluse ... 239
immutable objects ... 710
immutable properties ... 710
implicit contexts ... 571
importing stencils into text ... 702
`improvisationOff` ... 41, 76
`improvisationOn` ... 41, 76
improvvisazione ... 41
include-settings ... 486
including files ... 480
incorniciatura del testo ... 237
`indent` ... 200, 509, 542
indicating No Chord in ChordNames ... 398
indicating position and barring for fretted instruments ... 366
indicazione di tempo ... 61
indicazione di tempo, impostazioni predefinite ... 62
indicazione di tempo, stile ... 62
indicazione di tempo, visibilità dell' ... 62
indicazione manuale di ripetizione ... 151
indicazione metronomica ... 66
indicazione metronomica con testo ... 66
indicazioni di diteggiatura per accordi ... 212
indicazioni di tempo composto ... 74
indicazioni di tempo doppie ... 72
indicazioni di tempo polimetrico ... 72
indicazioni di tempo, ripristinare i valori predefiniti delle proprietà delle ... 64
indicazioni dinamiche multiple su una nota ... 119
indicazioni dinamiche nuove ... 124
indicazioni dinamiche, più di un segno su una nota ... 119
indicazioni polimetriche ... 72
indicazioni testuali ... 226
informazioni sul tempo e ripetizioni ... 150
inizializzazione del rigo ... 181
inizio del sistema ... 182
inizio ripetizione ... 151
inlining an Encapsulated PostScript image ... 680
`inner-margin` ... 508
inserting music into text ... 687
inserting PostScript directly into text ... 682
inserting URL links into text ... 684
`inStaffSegno` ... 745
instrument names ... 494
`instrumentSwitch` ... 201, 746
intavolatura ... 181
interface ... 710
interface, layout ... 572
Internals Reference ... 553
interruzioni di linea ... 93
interruzioni di linea, cadenze ... 72
interruzioni di linea, musica in tempo libero ... 72
interruzioni di linea, travature ... 79
interruzioni di pagina, cadenze ... 72
interruzioni di pagina, musica in tempo libero ... 72
interruzioni nella musica in tempo libero ... 72
`inversion` ... 746
inversione ... 13
inversione modale ... 15
invisibile, gambo ... 217
invisibili, note ... 214
`ionian` ... 21
ionio ... 21
`italic` ... 231

J

jazz chords ... 400
`justified-lines` ... 242
`justify` ... 236
justifying lines of text ... 703
justifying text ... 669

K

keep tagged music ... 483
`keepWithTag` ... 746
`key` ... 20, 39, 746
key signature ... 422, 426
key signature, visibility following explicit change ... 595
keyboard instrument staves ... 311
keyboard music, centering dynamics ... 311
keyed instrument staves ... 311
KievanStaffContext ... 435
KievanVoiceContext ... 435
`killCues` ... 209, 746
kirchenpausen ... 59

L

`label` ... 746
laissez vibrer ... 51
`laissezVibrer` ... 51
`language` ... 746
`languageRestore` ... 746
`languageSaveAndChange` ... 746
`large` ... 210
`large` ... 233
`larger` ... 232
`larger` ... 233
`last-bottom-spacing` ... 507
layers ... 593
`layout file` ... 514
layout interface ... 572
layout objects ... 709
left aligning text ... 670
`left-align` ... 234

`left-margin` ... 507
legatura di frase ... 126
legatura di frase puntata ... 129
legatura di frase tratteggiata ... 129
legatura di frase, definizione dei modelli di tratteggio ... 129
legatura di frase, metà continua e metà tratteggiata ... 129
legatura di portamento continua ... 126
legatura di portamento e ripetizioni ... 150
legatura di portamento punteggiata ... 126
legatura di portamento tratteggiata ... 126
legatura di portamento, definizione dei modelli di tratteggio per il fraseggio ... 129
legatura di portamento, definizione del modello di tratteggio ... 127
legatura di portamento, frase puntata ... 129
legatura di portamento, frase tratteggiata ... 129
legatura di portamento, fraseggio multiplo ... 129
legatura di portamento, fraseggio simultaneo ... 129
legatura di portamento, fraseggio, definizione dei modelli di tratteggio ... 129
legatura di portamento, metà tratteggiata e metà continua ... 127
legatura di portamento, tratto metà continuo e metà tratteggiato ... 129
legatura di valore ... 50
legatura di valore e alterazione ... 6
legatura di valore, laissez vibrer ... 51
legature di frase ... 129
legature di frase multiple ... 129
legature di frase simultanee ... 129
legature di portamento ... 126
legature di portamento multiple ... 126
legature di portamento simultanee ... 126
legature di portamento, posizionamento manuale ... 126
legature di portamento, sopra le note ... 126
legature di portamento, sotto le note ... 126
legature di portamento, stile ... 126
legature di valore e accordi ... 51
legature di valore e parentesi della volta ... 51
legature di valore punteggiate ... 52
legature di valore tratteggiate ... 52
legature di valore, aspetto ... 52
legature di valore, finali alternativi ... 146
legature di valore, nelle ripetizioni ... 146
legature di valore, posizionamento ... 52
legature di valore, ripetizione ... 51
lexer ... 710
lheel ... 706
lidio ... 21
Ligatures ... 415, 438
ligatures in text ... 663
LilyPond grammar ... 710
line breaks ... 515
line, cross-staff ... 315
line, staff-change ... 315
line, staff-change follower ... 315
`line-width` ... 507, 542
linee del rigo, fermare e avviare ... 188
linee del rigo, modificare ... 188
linee verticali tra i righi ... 219
lineprall ... 706
lingua, nomi delle altezze in un'altra ... 7
lingua, nomi delle note in un'altra ... 7
List of colors ... 631
`locrian` ... 21
locrio ... 21
`longa` ... 43, 54
longfermata ... 706
`lower` ... 234
lowering text ... 671
ltoe ... 706
lunghezza delle note ... 43
`ly:minimal-breaking` ... 519
`ly:one-line-breaking` ... 519
`ly:optimal-breaking` ... 518
`ly:page-turn-breaking` ... 518
`lydian` ... 21
lyrics and melodies ... 251
lyrics and tied notes ... 272
lyrics on grace notes ... 280
lyrics punctuation ... 248
lyrics, aligning to a melody ... 249
lyrics, aligning with sporadic melody ... 559
lyrics, divided ... 275
lyrics, entering ... 248
lyrics, formatting ... 248
Lyrics, increasing space between ... 265
lyrics, positioning ... 261
lyrics, repeating ... 267
lyrics, repeats with alternative endings ... 271
lyrics, shared among voices ... 276
lyrics, skipping notes ... 271
lyrics, using variables ... 259

M

`m` ... 394
maggiore ... 21
`magnify` ... 232
magnifying text ... 656
`magstep` ... 210, 586
`maj` ... 394
`major` ... 21
`major seven symbols` ... 402
`majorSevenSymbol` ... 401
makam ... 449
makamlar ... 449
`make-dynamic-script` ... 124
`make-pango-font-tree` ... 245
`makeClusters` ... 164, 746
`makeDefaultStringTuning` ... 746
manual staff changes ... 312
`Manuali` ... 1
maqam ... 444
maqams ... 444
marcato ... 115, 706
margine, testo che va oltre ... 224
`mark` ... 105, 226, 746
`markLengthOff` ... 67, 227
`markLengthOn` ... 67, 227
`markup` ... 226, 228, 229, 230
markup multilinea ... 236
markup, allineare ... 234
markup, centrare sulla pagina ... 236
markup, comandi di allineamento del testo ... 237
markup, conditional ... 467

markup, decorazione ... 237
markup, espressioni ... 230
markup, incorniciatura ... 237
markup, multipagina ... 242
markup, notazione musicale dentro ... 240
markup, padding ... 238
markup, partitura dentro ... 241
markup, sintassi ... 230
markup, testo a capo ... 236
markup, testo giustificato ... 236
`markup-markup-spacing` ... 507
`markup-system-spacing` ... 506
`markuplist` ... 229, 242, 243
`max-systems-per-page` ... 509
`maxima` ... 43, 54
`measureLength` ... 81, 114
`measurePosition` ... 69, 114
Medicaea, Editio ... 414
medium intervals ... 444
melisma ... 255, 259
melismata ... 255
melismi, con travature ... 79
melodia, mostrare i ritmi della ... 76
mensural ... 414
Mensural ligatures ... 423
mensurale, formattazione ... 185
`MensuralStaff` ... 181
MensuralStaffContext ... 417
MensuralVoiceContext ... 417
mensuration sign ... 419
`mergeDifferentlyDottedOff` ... 168
`mergeDifferentlyDottedOn` ... 168
`mergeDifferentlyHeadedOff` ... 168
`mergeDifferentlyHeadedOn` ... 168
merging text ... 663
mezzosoprano, chiave di ... 17
`mf` ... 118
Microtones in MIDI ... 494
microtoni ... 8
MIDI ... 24, 491
MIDI block ... 491
MIDI context definitions ... 492
MIDI, articulations ... 494
MIDI, chord names ... 494
MIDI, instruments ... 494
MIDI, microtones ... 494
MIDI, Pitches ... 494
MIDI, quarter tones ... 494
MIDI, Rhythms ... 494
`min-systems-per-page` ... 509
`minimum-Y-extent` ... 523
`minimumFret` ... 327, 363
`minimumPageTurnLength` ... 518
`minimumRepeatLengthForPageTurn` ... 519
`minor` ... 21
`minorChordModifier` ... 402
minore ... 21
mirroring markup ... 683
misolidio ... 21
misura ... 61
misura parziale ... 69
misura, numeri ... 100
misura, raggruppamenti ... 86
misura, ripetizioni ... 155
misura, sottoraggruppamenti ... 86
misura, stanghette ... 93
misura, stanghette manuali ... 94
`mixed` ... 317
`mixolydian` ... 21
modale, inversione ... 15
modale, trasposizione ... 14
modali, trasposizioni ... 14
`modalInversion` ... 15, 746
modalità markup, caratteri speciali ... 230
modalità markup, testo tra virgolette ... 230
`modalTranspose` ... 14, 746
mode ... 711
`modern` ... 28
modern, stile delle alterazioni ... 28, 29
`modern-cautionary` ... 29
modern-cautionary, stile delle alterazioni ... 28, 29
`modern-voice` ... 29
`modern-voice-cautionary` ... 29
modern-voice-cautionary, stile delle alterazioni ... 29
moderntab clef ... 341
modi ... 21
modi ecclesiastici ... 21
modificare gli abbellimenti ... 109
modificare i nomi degli strumenti ... 200
modifiers, in chords ... 393
mordent ... 706
mordente inferiore (\mordent) ... 115
mordente inferiore, giù ... 115
mordente inferiore, su ... 115
mordente superiore (\prall) ... 115
mordente superiore, giù ... 115
mordente superiore, su ... 115
movements, multiple ... 451
`mp` ... 118
multilinea, markup ... 236
multilinea, testo ... 236
`MultiMeasureRestText` ... 58
multipagina, testo ... 242
musica dentro il blocco markup ... 240
Musica ficta ... 422
musica in tempo libero ... 70, 114
musica in tempo libero, alterazioni ... 70
musica in tempo libero, interruzioni di linea ... 72
musica in tempo libero, interruzioni di pagina ... 72
musica in tempo libero, numeri di battuta ... 70
musica in tempo libero, stanghette ... 70
musica in tempo libero, travature ... 70
musica mensurale, trascrizione di ... 185
musica parallela ... 178
musica per principianti ... 37
musica polifonica ... 168
musica rinascimentale ... 185
musical cues ... 292
`musicglyph` ... 106
`musicMap` ... 746
musicologia, analisi ... 221
musicQuotes ... 711
mutable objects ... 710
mutable properties ... 710

N

N.C. symbol ... 398
`name` ... 568

name of singer ... 279
names, character ... 290
nascondere i righi ... 195
nascondere i righi antichi ... 197
nascondere i righi ritmici ... 197
nascoste, note ... 214
natural harmonics ... 322
`neo-modern` ... 30
neo-modern, stile delle alterazioni ... 30
`neo-modern-cautionary` ... 30
neo-modern-cautionary, stile delle alterazioni ... 30
`neo-modern-voice` ... 30
neo-modern-voice, stile delle alterazioni ... 30
`neo-modern-voice-cautionary` ... 31
neo-modern-voice-cautionary, stile delle alterazioni ... 31
neomensural ... 415
new contexts ... 555
new spacing area ... 540
niente, al ... 121
no chord symbol ... 398
`no-reset` ... 31
no-reset, stile delle alterazioni ... 31
`noBeam` ... 90
nomi degli strumenti ... 199
nomi degli strumenti abbreviati ... 199
nomi degli strumenti, aggiungerli ad altri contesti ... 200
nomi degli strumenti, centrare ... 199
nomi degli strumenti, complessi ... 199
nomi degli strumenti, modifica ... 200
nomi delle altezze ... 1
nomi delle altezze, altre lingue ... 7
nomi delle note predefiniti ... 5
nomi delle note, altre lingue ... 7
nomi delle note, olandese ... 5
nomi delle note, predefinito ... 5
non-ASCII characters ... 486
`nonstaff-nonstaff-spacing` ... 523
`nonstaff-relatedstaff-spacing` ... 523
`nonstaff-unrelatedstaff-spacing` ... 523
`noPageBreak` ... 746
`noPageTurn` ... 746
`normal-size-super` ... 232
`normalsize` ... 210
`normalsize` ... 233
nota spaziatrice ... 56
nota, durata predefinita ... 43
nota, spostamento ... 168
notazione dentro il blocco markup ... 240
notazione grafica ... 239
notazione semplificata ... 37
notazione, dimensione del tipo di carattere ... 210
notazione, spiegare la ... 218
note a forma variabile ... 38
note colorate ... 215
note colorate negli accordi ... 216
note doppiamente puntate ... 43
note fantasma ... 216
note head styles ... 653
note heads, ancient ... 420, 436
note heads, diamond-shaped ... 322
note in corpo più piccolo ... 202, 205
note invisibili ... 214
note nascoste ... 214
note più piccole ... 205
note puntate ... 43
note simultanee e alterazioni ... 32
note tra parentesi ... 216
note trasparenti ... 214
note, divisione ... 75
note, durata delle ... 43
note, lunghezza delle ... 43
note, trasposizione delle ... 10
note-event ... 204
`Note_heads_engraver` ... 75
notes within text by log and dot-count ... 685
notes within text by string ... 686
notes, cross-staff ... 312, 315
notes, spacing horizontally ... 540
notine ... 202, 205
notine, formattare le ... 205
`null` ... 234
NullVoice ... 276
numeri di battuta ... 100
numeri di battuta, cadenze ... 70
numeri di battuta, collisione ... 104
numeri di battuta, con lettere ... 102
numeri di battuta, con ripetizioni ... 102
numeri di battuta, controlli ... 104
numeri di battuta, disposizione a distanza regolare ... 100
numeri di battuta, musica in tempo libero ... 70
`numericTimeSignature` ... 62
numero del gruppo irregolare, modifiche del ... 46
numero della misura e ripetizioni ... 150
numero di battuta ... 114
numero di battuta, allineamento ... 103
numero di battuta, formato ... 101
numero di ripetizione, modificare ... 151
nuovo rigo ... 181
nuvoletta ... 218
nuvoletta di aiuto ... 218

O

objects, coloring ... 593
objects, overwriting ... 593
objects, rotating ... 598
objects, visibility of ... 592
`octaveCheck` ... 9, 746
`offset` ... 746
oggetti colorati ... 215
oggetti grafici incorporati ... 237
oggetti grafici, disegnare ... 237
oggetti grafici, includere ... 237
`omit` ... 747
on-the-fly ... 467
`once` ... 747
`oneVoice` ... 164
open ... 706
open string indication ... 321
operazione, inversione ... 13
operazione, inversione modale ... 15
operazione, retrogradazione ... 14
operazione, trasposizione ... 14
operazioni, modali ... 14
oratorio ... 286
orchestral strings ... 321
organo, segni del pedale ... 115

ornamenti 107
ossia 192, 197
ottava 23, 747
ottava assoluta 1
ottava relativa 2
ottava relativa e accordi 4
ottava relativa e trasposizione 5
ottava, controllo 9
ottavazione 23
Ottoman music 449
outer-margin 508
output definitions 553
output-count 711
output-def 710
output-suffix 711
outside-staff-horizontal-padding 537
outside-staff-padding 537
outside-staff-priority 537
overrideProperty 747
overrides, reverting 578
overrideTimeSignatureSettings 747
overriding for only one moment 578
overriding properties within text markup 700
overwriting objects 593

P

p 118
pad-around 238
pad-markup 238
pad-to-box 238
pad-x 238
padding 574
padding intorno al testo 238
padding text 671
padding text horizontally 672
page breaks 542
page layout 543
page numbers, auto-numbering 511
page numbers, specify the first 511
page numbers, suppress 511
page size 503
page, orientation 504
page-breaking 510
page-breaking-system-system-spacing 510
page-count 510
page-spacing-weight 511
pageBreak 747
pageTurn 747
palmMute 747
palmMuteOn 747
Pango 243
paper size 503
paper size, landscape 504
paper size, orientation 504
paper-height 505
paper-width 507
parallela, musica 178
parallelMusic 178, 747
parentesi 221
parentesi della volta 151
parentesi della volta con testo 152
parentesi della volta e legature di valore 51
parentesi di raggruppamento delle note 221
parentesi graffe, annidamento di 186
parentesi orizzontale 221
parentesi quadra verticale 182
parentesi quadre 216
parentesi quadre, annidamento di 186
parentesi uncinate (o angolari) 159
parentesi, fraseggio 221
parentesi, stile nell'arpeggio attraverso il rigo 140
parenthesize 216, 747
parlato 296
parlato, testa di nota 35
parser 710
parser variable 711
part songs 286
partcombine 173, 747
partcombineApart 174
partcombineAutomatic 174
partcombineChords 174
partcombineDown 748
partcombineForce 748
partCombineListener 711
partcombineSoloI 174
partcombineSoloII 174
partcombineUnisono 174
partcombineUp 748
parte a due 173
parte solista 173
partial 69, 748
partitura dentro il blocco markup 241
partitura senza i righi vuoti 195
parziale, misura 69
paths, drawing 681
pausa 54
pausa d'intero 54
pausa di breve 54
pausa di longa 54
pausa di maxima 54
pausa ecclesiastica 59
pausa intera per una misura intera 57
pausa invisibile 56
pausa multipla 54
pausa multipla con testo a margine 58
pausa multipla, attaccare fermata 58
pausa multipla, attaccare testo 58
pausa multipla, contrazione 58
pausa multipla, espansione 58
pausa multipla, script 58
pausa spaziatrice 56
pausa, collisioni di 61
pausa, inserire le durate 54
pausa, specificare la posizione verticale 54
pausa, spostamento automatico 168
pause d'intero 57
pause multiple 57
pause multiple e diteggiature 61
pause multiple, posizionamento 59
pause, condensare 61
pause, divisione 75
pedal diagrams, harp 319
pedal indication styles 317
pedal indication, bracket 317
pedal indication, mixed 317
pedal indication, text 317
pedal sustain style 317
pedal, sostenuto 317

pedal, sustain ... 317
pedale, segni ... 115
pedals, harp ... 319
pedals, piano ... 317
`pedalSustainStyle` ... 317
pedice ... 232
`percent` ... 155
percentuale, ripetizioni ... 155
percussion ... 370, 372
percussion clef ... 371
Petrucci ... 414, 415
phrasing, in lyrics ... 255
`phrasingSlurDashed` ... 129
`phrasingSlurDashPattern` ... 129, 748
`phrasingSlurDotted` ... 129
`phrasingSlurDown` ... 129
`phrasingSlurHalfDashed` ... 129
`phrasingSlurHalfSolid` ... 129
`phrasingSlurNeutral` ... 129
`phrasingSlurSolid` ... 129
`phrasingSlurUp` ... 129
`phrygian` ... 21
`piano` ... 29
piano e alterazioni ... 29, 30
piano music, centering dynamics ... 311
piano pedals ... 317
piano staves ... 311
piano, stile delle alterazioni ... 29
`piano-cautionary` ... 30
piano-cautionary, stile delle alterazioni ... 30
pianoforte, rigo per ... 182
`PianoStaff` ... 311, 313
`Pitch_squash_engraver` ... 76
`pitchedTrill` ... 141, 748
Pitches in MIDI ... 494
pitchnames ... 711
pizzicato, Bartók ... 323
pizzicato, snap ... 323
placement of lyrics ... 261
placing horizontal brackets around text ... 680
placing parentheses around text ... 681
placing vertical brackets around text ... 677
`pointAndClickOff` ... 748
`pointAndClickOn` ... 748
`pointAndClickTypes` ... 748
polifonia su un rigo singolo ... 164
polimetriche, indicazioni ... 72
pollice, indicazione ... 212
pollice, segno del (\thumb) ... 115
polymetric scores ... 562
polyphony, shared lyrics ... 276
portamenti indeterminati verso il basso (cadute) e verso l'alto ... 132
portato ... 115, 706
posizionamento della parentesi quadra del gruppo irregolare ... 45
posizionamento verticale delle dinamiche ... 120
posizionare pause multiple ... 59
`postscript` ... 239
power chords ... 368
`powerChords` ... 368
`pp` ... 118
`ppp` ... 118
`pppp` ... 118
`ppppp` ... 118
prall ... 706
pralldown ... 706
prallmordent ... 706
prallprall ... 706
prallup ... 706
predefined string tunings for fretted instruments ... 339
`predefinedFretboardsOff` ... 362
`predefinedFretboardsOn` ... 362
prima volta ... 143
principianti, musica ... 37
`print-all-headers` ... 511
`print-first-page-number` ... 511
`print-page-number` ... 511
printing chord names ... 398
printing order ... 593
prob ... 711
properties ... 575
properties, grob ... 577
property object ... 711
proprietà di disposizione automatica delle travature per le indicazioni di tempo ... 62
psalms ... 297
pull off ... 338
punctuation in lyrics ... 248
punto, note ... 43
pure containers, Scheme ... 611
`pushToTag` ... 748
putting space around text ... 671

Q

Quarter tones in MIDI ... 494
quarto di tono ... 6
`quotedCueEventTypes` ... 204
`quotedEventTypes` ... 204
`quoteDuring` ... 202, 205, 748
quotes in lyrics ... 248
quotes, in lyrics ... 255

R

`r` ... 54
`R` ... 57
`ragged-bottom` ... 505
`ragged-last` ... 508, 542
`ragged-last-bottom` ... 505
`ragged-right` ... 508, 542
raggruppamento dei gruppi irregolari ... 45
`raise` ... 234
raising text ... 672
rallentando in MIDI ... 494
Ratisbona, Editio ... 414
referencing contexts ... 555
referencing page labels in text ... 703
referencing page numbers in text ... 701
regular line breaks ... 516
relativa, ottava ... 2
`relative` ... 2, 5, 13, 314, 748
relative music and autochange ... 314
relativo ... 2
religious music ... 297
removals, in chords ... 396
remove tagged music ... 483

`RemoveEmptyStaves` ... 751
`removeWithTag` ... 748
`repeatCommands` ... 151
repeating lyrics with alternative endings ... 271
repeats and lyrics ... 267
repeats in MIDI ... 495
`repeatTie` ... 51
repetition, using `q` ... 328
`resetRelativeOctave` ... 748
respiri ... 130
`rest` ... 54
rest-event ... 204
`restrainOpenStrings` ... 327
rests or multi-measure-rests within text by log and dot-count ... 686
rests or multi-measure-rests within text by string ... 687
rests, ancient ... 421
retrogradazione, trasformazione ... 14
`retrograde` ... 14, 748
reverseturn ... 706
reverting overrides ... 578
`revertTimeSignatureSettings` ... 748
`rfz` ... 118
rgb, colore ... 215
`rgb-color` ... 215
rheel ... 706
`RhythmicStaff` ... 181
Rhythms in MIDI ... 494
ricopiate, ripetizioni ... 153
ridimensionamento dei righi ... 192
righi annidati ... 186
righi, gruppo di ... 182
right aligning text ... 673
right hand fingerings for fretted instruments ... 364
`right-align` ... 234
`right-margin` ... 508
`rightHandFinger` ... 364
`rightHandFinger` ... 749
rigo Gregoriano per trascrizione ... 181
rigo multiplo ... 182
rigo per batteria ... 181
rigo per coro ... 182
rigo per intavolatura ... 181
rigo per percussioni ... 181
rigo per pianoforte ... 182
rigo ritmico ... 181
rigo temporaneo ... 192, 195
rigo vuoto ... 195
rigo, batteria ... 181
rigo, nascondere ... 195
rigo, nuovo ... 181
rigo, percussioni ... 181
rigo, ridimensionamento del ... 192
rigo, simbolo del ... 188
rigo, singolo ... 181
rinascimentale, musica ... 185
ripetere le legature di valore ... 51
ripetizione con anacrusi ... 144
ripetizione con finali alternativi ... 143
ripetizione e numero della misura ... 150
ripetizione normale ... 143
ripetizione, breve ... 155
ripetizione, fine ... 151
ripetizione, inizio ... 151
ripetizione, manuale ... 151
ripetizione, tremolo ... 157
ripetizione, uso di `q` ... 161
ripetizioni alternate ... 153
ripetizioni annidate ... 150
ripetizioni con legature di valore ... 146
ripetizioni con percentuale ... 155
ripetizioni della misura ... 155
ripetizioni e glissandi ... 150
ripetizioni e informazioni sul tempo ... 150
ripetizioni e legatura di portamento ... 150
ripetizioni ricopiate ... 153
ripetizioni, alternativa ... 153
ripetizioni, ambiguità ... 150
ripetizioni, con segno ... 146
ripetizioni, numeri di battuta alternativi ... 149
ripetizioni, numeri di battuta con lettere ... 149
ripetizioni, unfold ... 153
ripristinare le proprietà predefinite delle indicazioni di tempo ... 64
ritmi di accompagnamento per chitarra, mostrare ... 76
ritmi di accompagnamento, mostrare ... 76
ritmi, mostrare la melodia ... 76
ritornelli ... 96
root of chord ... 393
rotating objects ... 598
rotating text ... 673
`rounded-box` ... 237
rtoe ... 706

S

`s` ... 56
Sacred Harp, testa di nota ... 38
`sacredHarpHeads` ... 38
`sacredHarpHeadsMinor` ... 39
SATB ... 286
scalable vector graphics output ... 490
scalare le durate ... 49
`scaleDurations` ... 50, 72, 749
scaling markup ... 683
scaling text ... 674
scelta della dimensione del tipo di carattere (per gli elementi della notazione) ... 210
Scheme object ... 711
Scheme variable ... 711
Scheme, pure containers ... 611
Scheme, unpure containers ... 611
`score-markup-spacing` ... 506
`score-system-spacing` ... 506
`scoreTitleMarkup` ... 464
Scottish highland bagpipe ... 382
script ... 115
script su pausa multipla ... 58
scritta ... 223
scrivere la musica in parallelo ... 178
seconda volta ... 143
segni del pedale ... 115
segni del pedale dell'organo ... 115
segni di chiamata ... 105
segni di tempo all'interno delle parentesi di un gruppo irregolare ... 49
segni di tremolo ... 158
segno ... 95, 106, 115, 706

segno del pollice (\thumb) ... 115
segno di bequadro ... 5
segno di chiamata manuale ... 106
segno di chiamata personalizzato ... 106
segno di chiamata, formato ... 106
segno di chiamata, stile ... 106
segno di modifica dell'ottava ... 1
segno di pausa ... 130
segno di spunta ... 131
segno separatore del sistema ... 187
segno sulla stanghetta ... 226
segno, con ripetizioni ... 146
segno, di chiamata, formato ... 106
segno, di chiamata, stile ... 106
`self-alignment-X` ... 523
Semai form ... 447
semi-bemolle ... 6, 8
semi-diesis ... 6, 8
Semi-flat symbol appearance ... 445
semicirculus ... 706
sesqui-bemolle ... 8
sesqui-diesis ... 8
`set` ... 81
`set-octavation` ... 23
setting extent of text objects ... 703
setting horizontal text alignment ... 666
setting subscript in standard font size ... 657
setting superscript in standard font size ... 657
`settingsFrom` ... 749
seventh chords ... 393
`sf` ... 118
`sff` ... 118
`sfz` ... 118
`shape` ... 749
shaping slurs and ties ... 606
shared properties ... 710
`shiftDurations` ... 749
`shiftOff` ... 168
`shiftOn` ... 168
`shiftOnn` ... 168
`shiftOnnn` ... 168
`short-indent` ... 200, 509
shortfermata ... 706
`show-available-fonts` ... 245
`showFirstLength` ... 489
showFirstLength ... 711
`showKeySignature` ... 382
`showLastLength` ... 489
showLastLength ... 711
`showStaffSwitch` ... 315
signumcongruentiae ... 706
simboli non musicali ... 239
simboli speciali di arpeggio ... 137
simbolo del rigo ... 188
simple closure ... 709
simple text strings ... 659
simple text strings with tie characters ... 689
sincronizzazione degli abbellimenti ... 111
singer name ... 279
`single` ... 749
sintassi di markup ... 230
sistema ... 182
sistema, delimitatori di inizio annidati ... 186
sistema, segno separatore ... 187
`skip` ... 56, 749
skipping notes in lyrics ... 271
`skipTypesetting` ... 489
`slashChordSeparator` ... 402
slashed digits ... 701
`slashedGrace` ... 749
slides in tablature notation ... 336
slur-event ... 204
`slurDashed` ... 126
`slurDashPattern` ... 127, 749
`slurDotted` ... 126
`slurDown` ... 126
`slurHalfDashed` ... 127
`slurHalfSolid` ... 127
`slurNeutral` ... 126
slurs, modifying ... 605
`slurSolid` ... 126
`slurUp` ... 127
`small` ... 210
`small` ... 233
`smaller` ... 232
`smaller` ... 233
smob ... 711
snap pizzicato ... 323
snappizzicato ... 706
Sol, chiave di ... 17
Solesmes ... 414
soprano, chiave di ... 17
sos ... 317
sostenuto pedal ... 317
`sostenutoOff` ... 317
`sostenutoOn` ... 317
Sound ... 491
Southern Harmony, testa di nota ... 38
`southernHarmonyHeads` ... 38
`southernHarmonyHeadsMinor` ... 39
`sp` ... 118
space between staves ... 523
space inside systems ... 523
spaces in lyrics ... 248
spaces, in lyrics ... 255
`spacing` ... 539
spacing area, new ... 540
Spacing lyrics ... 265
spacing, display of layout ... 549
spacing, horizontal ... 538
spacing, vertical ... 523
`spacingTweaks` ... 749
`Span_stem_engraver` ... 315
spanners, modifying ... 609
special characters ... 486
spezzato, arpeggio ... 137
splice into tagged music ... 483
spostamento automatico della pausa ... 168
spostare le voci ... 168
spostare una nota ... 168
`spp` ... 118
Sprechgesang ... 296
spunta, segno di ... 131
Square neumes ligatures ... 428
staccatissimo ... 115, 706
staccato ... 115, 706
stacking text in a column ... 663
staff change line ... 315
staff changes, automatic ... 313
staff changes, manual ... 312

staff distance ... 523
staff size, setting ... 514
staff switching ... 315
staff symbol, setting of ... 586
`staff-affinity` ... 523
staff-change line ... 315
`staff-staff-spacing` ... 523
`Staff.midiInstrument` ... 494
`Staff_symbol_engraver` ... 195
`staffgroup-staff-spacing` ... 523
StaffSymbol ... 188
stampare i caratteri riservati ... 230
stampare i caratteri speciali ... 230
stanghette ... 93
stanghette di chiusura ... 93
stanghette doppie ... 93
stanghette invisibili ... 93
stanghette manuali ... 94
stanghette predefinite, modifica ... 99
stanghette, cadenze ... 70
stanghette, definire ... 97
stanghette, musica in tempo libero ... 70
stanghette, simboli sulle ... 226
stanza number ... 278
`start-repeat` ... 151
`startGroup` ... 221
`startStaff` ... 188, 192
`startTrillSpan` ... 140
staves, keyboard instruments ... 311
staves, keyed instruments ... 311
staves, piano ... 311
`Stem` ... 315
`stem-spacing-correction` ... 539
`stemDown` ... 217
`stemLeftBeamCount` ... 90
`stemNeutral` ... 217
`stemRightBeamCount` ... 90
stems, cross-staff ... 315
`stemUp` ... 217
stencil ... 711
stencil, removing ... 592
stile del segno di chiamata ... 106
stile delle alterazioni *default* ... 28
stile delle alterazioni di precauzione *modern voice* ... 29
stile delle alterazioni *dodecaphonic* ... 31
stile delle alterazioni *forget* ... 31
stile delle alterazioni *modern* ... 28, 29
stile delle alterazioni *modern-cautionary* ... 28
stile delle alterazioni *modern-voice-cautionary* ... 29
stile delle alterazioni *neo-modern* ... 30
stile delle alterazioni *neo-modern-cautionary* ... 30
stile delle alterazioni *neo-modern-voice-cautionary* ... 31
stile delle alterazioni *no-reset* ... 31
stile delle alterazioni *piano* ... 29
stile delle alterazioni *piano-cautionary* ... 30
stile delle alterazioni *teaching* ... 31
stile delle alterazioni *voice* ... 28
stile delle alterazioni, *neo-modern-voice* ... 30
stile di alterazione ... 26
stile di alterazione predefinito ... 26
stile moderno delle alterazioni ... 28
stile, legatura di portamento ... 126
stili delle teste di nota ... 35
stili di voce ... 168
`stopGroup` ... 221
stopped ... 706
`stopStaff` ... 188, 192, 195
`stopTrillSpan` ... 140
`storePredefinedDiagram` ... 355, 749
string numbers ... 324
string vs. fingering numbers ... 324
string, indicating open ... 321
strings, orchestral ... 321
strings, writing for ... 321
`stringTuning` ... 340, 749
`stringTunings` ... 339, 351
strumenti traspositori ... 11
strumenti, nomi complessi ... 199
strumenti, nomi degli ... 199
strumento trasportore ... 24
strumento, cambio di ... 201
`styledNoteHeads` ... 749
`sub` ... 232
subbasso, chiave di ... 17
subscript text ... 660
suddivisioni, raggruppamenti ... 86
`suggestAccidentals` ... 422
`super` ... 232
superscript text ... 660
`sus` ... 396
sustain pedal ... 317
sustain pedal style ... 317
`sustainOff` ... 317
`sustainOn` ... 317
SVG output ... 490
syllable durations, automatic ... 251
`system-count` ... 509
`system-separator-markup` ... 511
`system-system-spacing` ... 507
`systems-per-page` ... 509

T

tab clef ... 341
`tabChordRepeats` ... 749
`tabChordRepetition` ... 749
tablatura ... 181
tablature ... 324
tablature and harmonic indications ... 330
tablature and slides ... 336
tablature, banjo ... 324, 339, 369
tablature, bass ... 339
tablature, bass guitar ... 339
tablature, cello ... 339
tablature, custom string tunings ... 340
tablature, double bass ... 339
tablature, guitar ... 324, 339
tablature, mandolin ... 339
tablature, predefined string tunings ... 339
tablature, ukulele ... 339
tablature, viola ... 339
tablature, violin ... 339
tablatures, basic ... 326
tablatures, custom ... 339
tablatures, default ... 326
`TabStaff` ... 181, 326
`TabVoice` ... 326

tag ... 483
`tag` ... 750
tagli addizionali ... 188
tagli addizionali, funzionamento interno ... 188
tagli addizionali, modificare ... 188
tagliata, testa di nota ... 41
tante voci ... 168
`taor` ... 382
taqasim ... 447
`teaching` ... 31
teaching, stile delle alterazioni ... 31
`teeny` ... 210
`teeny` ... 233
tempi polimetrici, con travature ... 72
Template Arabic music ... 447
tempo ... 61
`tempo` ... 66
tempo (all'interno della partitura) ... 114
tempo composto, indicazioni ... 74
tempo, polimetrico ... 72
tempo, stile ... 62
`temporary` ... 750
tenore, chiave di ... 17
tenuto ... 115, 706
terzina, formattazione della ... 46
terzine ... 45
testa di nota tagliata ... 41
testa di nota, Aiken ... 38
testa di nota, Christian Harmony ... 38
testa di nota, forma ... 38
testa di nota, Funk ... 38
testa di nota, Harmonica Sacra ... 38
testa di nota, improvvisazione ... 41
testa di nota, Sacred Harp ... 38
testa di nota, Southern Harmony ... 38
testa di nota, Walker ... 38
teste di nota ... 210
teste di nota a rombo ... 35
teste di nota barrate ... 35
teste di nota facili da suonare ... 37
teste di nota speciali ... 35
teste di nota, armonico ... 35
teste di nota, chitarra ... 35
teste di nota, esercizio ... 37
teste di nota, notazione semplificata ... 37
teste di nota, parlato ... 35
teste di nota, stili ... 35
testo a capo automaticamente ... 236
testo a margine ... 230
testo al livello superiore ... 228
testo assegnato a una voce ... 164
testo esteso su più pagine ... 242
testo formattato su più pagine ... 242
testo fuori dal margine ... 224
testo giustificato ... 236
testo in colonne ... 236
testo indipendente ... 228
testo multilinea ... 236
testo nella parentesi della volta ... 152
testo separato ... 228
testo su pausa multipla ... 58
testo sulla stanghetta ... 226
testo tra virgolette ... 223
testo tra virgolette in modalità markup ... 230
testo vocale, tenerlo dentro il margine ... 224
testo, \skip ... 56
testo, allineamento ... 234
testo, allineamento orizzontale ... 234
testo, allineamento verticale ... 234
testo, altre lingue ... 223
testo, centrare sulla pagina ... 236
testo, comandi di allineamento del ... 237
testo, con travature ... 81
testo, decorazione ... 237
testo, dimensione ... 232
testo, incorniciatura ... 237
testo, indicazioni ... 226
testo, padding ... 238
testo, tenerlo dentro il margine ... 224
`text` ... 317
text columns, left-aligned ... 670
text columns, right-aligned ... 673
`textLengthOff` ... 59, 223
`textLengthOn` ... 59, 223
`textSpannerDown` ... 224
`textSpannerNeutral` ... 224
`textSpannerUp` ... 224
Thorough bass ... 406
`thumb` ... 212
thumb ... 706
`tieDashed` ... 52
`tieDashPattern` ... 750
`tieDotted` ... 52
`tieDown` ... 52
`tieNeutral` ... 52
ties, in lyrics ... 255
ties, modifying ... 605
`tieSolid` ... 52
`tieUp` ... 52
`time` ... 61, 81, 750
time signatures ... 419
Time signatures, multiple ... 562
`times` ... 750
`timeSignatureFraction` ... 72
`tiny` ... 210
`tiny` ... 233
tipi di carattere ... 243
tipi di carattere disponibili, elenco ... 245
tipi di carattere, cambiare ... 231
tipi di carattere, famiglie ... 233
tipi di carattere, modificarli per l'intero documento ... 245
tipi di carattere, trovare quelli disponibili ... 245
tipo di carattere standard (per gli elementi della notazione) ... 211
tipo di carattere, dimensione ... 232
tipo di carattere, ridimensionamento ... 210
`tocItem` ... 750
togliere le citazioni in corpo più piccolo ... 209
`Top` ... 1
`top-margin` ... 505
`top-markup-spacing` ... 507
`top-system-spacing` ... 507
toplevel-bookparts ... 711
toplevel-scores ... 711
`translate` ... 235
`translate-scaled` ... 235
translating text ... 674
transparent, making objects ... 592
`transpose` ... 5, 10, 13, 750

transposed clefs, visibility of ... 597
transposedCueDuring ... 208, 750
transposing fret diagrams ... 353
transposition ... 24, 202, 750
trascrizione di musica mensurale ... 185
trasformazione retrograda ... 14
trasparenti, note ... 214
trasporre ... 10
traspositori, strumenti ... 11
trasposizione ... 10
trasposizione dell'ottava ... 18
trasposizione delle altezze ... 10
trasposizione delle note ... 10
trasposizione e ottava relativa ... 5
trasposizione MIDI ... 24
trasposizione modale ... 14
trasposizione opzionale dell'ottava ... 18
trasposizione, chiave ... 18
trasposizione, MIDI ... 24
trasposizione, strumento ... 24
trasposizioni modali ... 14
tratti di suddivisione della travatura, direzione ... 86
travatura, direzione dei tratti di suddivisione della ... 86
travatura, estremità in una partitura ... 87
travatura, estremità in voci multiple ... 87
travature a raggiera ... 92
travature angolari ... 79
travature con angolazione ... 79
travature convergenti o divergenti ... 92
travature del tremolo ... 157
travature manuali ... 78, 89
travature manuali, abbellimenti ... 89
travature manuali, scorciatoia per impostare la direzione ... 89
travature, \partcombine con \autoBeamOff ... 79
travature, cadenze ... 70
travature, con melismi ... 79
travature, con tempi polimetrici ... 72
travature, con testo ... 81
travature, interruzioni di linea ... 79
travature, musica in tempo libero ... 70
travature, personalizzazione delle regole ... 78
travature, suddivisione ... 85
tre corde ... 317
treCorde ... 317
tremolo ... 157
tremolo tra due righi ... 159
tremolo, segni ... 158
tremoloFlags ... 158
triads ... 393
triangle ... 239
trill ... 140
trill ... 706
trilli ... 140
trilli con alterazione ... 141
trilli con altezza ... 141
trilli con notina ... 141
trilli con notina e alterazione ... 141
trillo (\trill) ... 115
trills in MIDI ... 494
trovare i tipi di carattere disponibili ... 245
tuning, non-Western ... 443
tunings, banjo ... 370
tuplet ... 45
tuplet ... 72, 750
tupletDown ... 45
tupletNeutral ... 45
TupletNumber ... 46
tupletNumberFormatFunction ... 46
tupletSpan ... 750
tupletSpannerDuration ... 46
tupletUp ... 45
Turkish music ... 449
Turkish note names ... 449
turn ... 706
turns in MIDI ... 494
tweak ... 750
tweak, relation to \override ... 581
tweaking ... 579
tweaking control points ... 581
two-sided ... 508
type ... 568
typeface ... 709

U

U.C. ... 317
ukulele ... 343
una corda ... 317
unaCorda ... 317
underline ... 231
underlining text ... 661
undo ... 751
unfold ... 153
unfold, finali alternativi ... 153
unfold, ripetizione ... 153
unfoldRepeats ... 751
unHideNotes ... 214
Unicode ... 487
unione delle parti ... 173
unire le note ... 168
unpure containers, Scheme ... 611
up bow indication ... 321
upbow ... 706
upmordent ... 706
upprall ... 706
UTF-8 ... 486

V

varbaritono, chiave di ... 17
varcoda ... 115, 706
variables ... 455
variables, use of ... 482
Vaticana, Editio ... 414
VaticanaStaff ... 181
VaticanaStaffContext ... 424
VaticanaVoiceContext ... 424
vertical spacing ... 523, 543
VerticalAxisGroup ... 523
vertically centering text ... 674
verylongfermata ... 706
violino, chiave di ... 17
visibility of objects ... 592
visibility of transposed clefs ... 597
voce ... 164
voci multiple ... 168
voci, \partcombine con \autoBeamOff ... 79

voci, citare le 202, 205
voci, stili 168
`voice` 26, 28
`Voice` 164
voice, following 315
voice, stile delle alterazioni 28
`voiceOne` 164
voices, divided 288
`void` 751
volta 143
volta della ripetizione, modificare 151
volta, parentesi 151
volta, prima 143
volta, seconda 143

W

Walker, testa di nota 38
`walkerHeads` 38
`walkerHeadsMinor` 39
`whichBar` 99
White mensural ligatures 423
whitespace 455
wind instruments 379
`with-color` 215
`withMusicProperty` 751
`wordwrap` 236
`wordwrap-lines` 242

X

`X-offset` 523
x11, colore 215
`x11-color` 215, 216
`xNote` 751
`xNotesOn` 751