# LilyPond

Il compositore tipografico per la musica

# Utilizzo

## Il team di sviluppo di LilyPond

Questo manuale spiega come eseguire i programmi distribuiti con LilyPond versione 2.17.26. Inoltre, suggerisce alcune delle "migliori pratiche" per un uso efficiente.

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 2.17.26

# Sommario


**1 Eseguire `lilypond` ... 1**

1.1 Uso normale ... 1
1.2 Uso da linea di comando ... 1
Utilizzo di `lilypond` ... 1
Comandi standard da shell ... 1
Opzioni di base della linea di comando per LilyPond ... 2
Opzioni avanzate della linea di comando per `lilypond` ... 4
Variabili d'ambiente ... 8
LilyPond in una gabbia chroot ... 9
1.3 Messaggi di errore ... 10
1.4 Errori comuni ... 11
La musica esce dalla pagina ... 11
Appare un rigo in più ... 12
Messaggio di errore Unbound variable % ... 13
Messaggio di errore FT_Get_Glyph_Name ... 13
Avvertimento sul fatto che le affinità del rigo devono solo diminuire ... 13
Messaggio di errore `\new` inaspettato ... 13

**2 Aggiornare i file con `convert-ly` ... 15**

2.1 Perché la sintassi cambia? ... 15
2.2 Utilizzo di `convert-ly` ... 15
2.3 Opzioni da linea di comando per `convert-ly` ... 16
2.4 Problemi nell'eseguire `convert-ly` ... 16
2.5 Conversioni manuali ... 17

**3 Eseguire `lilypond-book` ... 19**

3.1 Un esempio di documento musicologico ... 19
3.2 Integrare musica e testo ... 23
3.2.1 LaTeX ... 23
3.2.2 Texinfo ... 25
3.2.3 HTML ... 26
3.2.4 DocBook ... 26
3.3 Opzioni dei frammenti musicali ... 27
3.4 Utilizzo di `lilypond-book` ... 30
3.5 Estensioni dei nomi di file ... 33
3.6 Modelli per lilypond-book ... 33
3.6.1 LaTeX ... 33
3.6.2 Texinfo ... 34
3.6.3 html ... 34
3.6.4 xelatex ... 35
3.7 Condividere l'indice ... 36
3.8 Metodi alternativi per combinare testo e musica ... 37

**4 Programmi esterni ... 38**

4.1 Punta e clicca ... 38
4.1.1 Configurare il sistema ... 38
Usare Xpdf ... 38
Usare GNOME 2 ... 38
Usare GNOME 3 ... 39
Ulteriore configurazione per Evince ... 39
Abilitare il punta e clicca ... 39
Punta e clicca selettivo ... 40
4.2 LilyPond e gli editor di testo ... 40
Modalità di Emacs ... 40
Modalità di Vim ... 41
Altri editor ... 41
4.3 Conversione da altri formati ... 41
4.3.1 Utilizzo di `midi2ly` ... 41
4.3.2 Utilizzo di `musicxml2ly` ... 43
4.3.3 Utilizzo di `abc2ly` ... 44
4.3.4 Utilizzo di `etf2ly` ... 44
4.3.5 Altri formati ... 45
4.4 Inclusione di partiture LilyPond in altri programmi ... 45
Tante citazioni da una grande partitura ... 45
Inserire l'output di LilyPond in OpenOffice.org ... 45
Inserire l'output di LilyPond in altri programmi ... 45
4.5 `include` indipendenti ... 46
4.5.1 Articolazione MIDI ... 46

**5 Consigli su come scrivere i file ... 47**

5.1 Consigli generali ... 47
5.2 Scrivere musica esistente ... 48
5.3 Grandi progetti ... 48
5.4 Risoluzione dei problemi ... 49
5.5 Make e Makefile ... 49

**Appendice A GNU Free Documentation License ... 56**

**Appendice B Indice di LilyPond ... 63**

# 1 Eseguire lilypond

Questo capitolo descrive dettagliatamente gli aspetti tecnici dell'esecuzione di LilyPond.

## 1.1 Uso normale

La maggior parte degli utenti esegue LilyPond attraverso un'interfaccia grafica (GUI); se non lo hai già fatto, leggi il Sezione "Tutorial" in *Manuale di Apprendimento*. Se usi un editor diverso per scrivere i file lilypond, leggi la documentazione di quel programma.

## 1.2 Uso da linea di comando

Questa sezione contiene informazioni aggiuntive sull'uso di LilyPond da linea di comando. Questo può essere utile per assegnare opzioni aggiuntive al programma. Inoltre, ci sono alcuni programmi complementari di 'aiuto' (come `midi2ly`) che funzionano solo da linea di comando.

Con 'linea di comando' si intende la linea di comando del sistema operativo. Gli utenti Windows avranno più familiarità con i termini 'shell DOS' o 'shell dei comandi'. Gli utenti MacOS X avranno più familiarità con i termini 'terminale' o 'console'. Una configurazione ulteriore è necessaria per gli utenti MacOS X; si veda Sezione "MacOS X" in *Informazioni generali*.

Descrivere come usare questa parte di un sistema operativo non rientra negli obiettivi di questo manuale; si prega di consultare altra documentazione su questo argomento se non si conosce la linea di comando.

### Utilizzo di lilypond

L'eseguibile `lilypond` può essere lanciato dalla linea di comando nel seguente modo.

```
lilypond [opzione]... file...
```

Se invocato con un nome di file senza estensione, viene tentata per prima l'estensione '`.ly`'. Per leggere l'input da stdin, usare un trattino (`-`) al posto di *file*.

Quando '`file.ly`' viene elaborato, lilypond creerà '`file.ps`' e '`file.pdf`' come output. Possono essere specificati molti file; ognuno di essi sarà elaborato in modo indipendente.[1]

Se '`file.ly`' contiene più di un blocco `\book`, allora tutte le altre partiture verranno salvate in file numerati, a partire da '`file-1.pdf`'. Inoltre, il valore di `output-suffix` (suffisso di output) sarà inserito tra la base del nome del file e il numero. Un file di input che contiene

```
#(define output-suffix "violin")
\score { ... }
#(define output-suffix "cello")
\score { ... }
```

produrrà come output *base*'`-violin.pdf`' e *base*'`-cello-1.pdf`'.

### Comandi standard da shell

Se la shell (ovvero la finestra dei comandi) utilizzata supporta le normali redirezioni, potrebbe essere utile usare i seguenti comandi per dirigere l'output di una console in un file:

- `lilypond file.ly 1>stdout.log` per redirigere l'output normale
- `lilypond file.ly 2>stderr.log` per redirigere i messaggi di errore
- `lilypond file.ly &>all.log` per redirigere tutto l'output

Consulta la documentazione della tua shell per vedere se supporta queste opzioni o se la sintassi è diversa. Nota che questi sono comandi shell e non hanno niente a che fare con lilypond.

[1] Lo status di GUILE non viene resettato dopo l'elaborazione di un file `.ly`: attenzione a non cambiare alcun valore predefinito dall'interno di Scheme.

## Opzioni di base della linea di comando per LilyPond

Sono contemplate le seguenti opzioni:

`-d,--define-default=`*`variabile=valore`*

Si veda ⟨undefined⟩ [Advanced command line options for LilyPond], pagina ⟨undefined⟩.

`-e,--evaluate=`*`espressione`*

Valuta l'*espressione* di Scheme prima di analizzare qualsiasi file '`.ly`'. Si possono specificare varie opzioni '`-e`'; saranno analizzate in modo sequenziale.

L'espressione sarà analizzata nel modulo `guile-user`, dunque se vuoi usare delle definizioni in *espressione*, usa

```
lilypond -e '(define-public a 42)'
```

nella linea di comando, e includi

```
#(use-modules (guile-user))
```

in cima al file `.ly`.

> **Nota:** Gli utenti Windows devono usare i doppi apici invece dei singoli apici.

`-f,--format=`*`formato`*

Formati di output. Come `formato` si può scegliere tra `ps`, `pdf` e `png`.

Esempio: `lilypond -fpng` *`file`*`.ly`

`-h,--help`

Mostra una sintesi dell'utilizzo.

`-H,--header=`*`CAMPO`*

Estrae un campo dell'intestazione nel file '`NOME.`*`CAMPO`*'.

`-i,--init=`*`file`*

Imposta il file di inizializzazione su *file* (predefinito: '`init.ly`').

`-I, --include=`*`directory`*

Aggiunge *directory* al percorso di ricerca per i file di input.

È possibile assegnare più opzioni -I. La ricerca inizierà nella prima directory definita, e se il file da includere non viene trovato la ricerca continuerà nelle directory seguenti.

`-j,--jail=`*`utente,gruppo,gabbia,directory`*

Esegue `lilypond` in una gabbia chroot.

L'opzione '`--jail`' fornisce un'alternativa più flessibile a '`--safe`' quando la formattazione di LilyPond è messa a disposizione attraverso un server web o quando LilyPond esegue sorgenti provenienti dall'esterno (si veda ⟨undefined⟩ [Advanced command line options for LilyPond], pagina ⟨undefined⟩).

L'opzione '`--jail`' modifica la radice di `lilypond` in *gabbia* appena prima di iniziare il vero processo di compilazione. L'utente e il gruppo vengono poi modificati per corrispondere a quelli forniti, e la directory corrente viene spostata in *directory*. Questa configurazione garantisce che non sia possibile (almeno in teoria) uscire dalla gabbia. Si noti che perché '`--jail`' funzioni `lilypond` deve essere eseguito come root; di solito questo si fa in modo sicuro col comando `sudo`.

Configurare una gabbia è una questione un po' delicata, perché bisogna essere sicuri che LilyPond possa trovare tutto quello di cui ha bisogno per compilare il sorgente *dentro la gabbia*. Una configurazione tipica comprende i seguenti elementi:

Impostare un filesystem distinto
: Si dovrebbe creare un filesystem separato LilyPond, così che possa essere montato con opzioni di sicurezza come `noexec`, `nodev`, e `nosuid`. In questo modo è impossibile lanciare degli eseguibili o scrivere su un dispositivo direttamente da LilyPond. Se non si vuole creare una partizione separata, si può creare un file di dimensioni ragionevoli e usarlo per montare un dispositivo di loop. Un filesystem separato garantisce inoltre che LilyPond non possa scrivere su uno spazio maggiore di quanto permesso.

Impostare un altro utente
: Per eseguire LilyPond in una gabbia si dovrebbe usare un altro utente e gruppo (ad esempio, `lily/lily`) con pochi privilegi. Ci dovrebbe essere una sola directory scrivibile da questo utente, che dovrebbe essere passata in *dir*.

Preparare la gabbia
: LilyPond ha bisogno di leggere alcuni file quando viene lanciato. Tutti questi file devono essere copiati nella gabbia, sotto lo stesso percorso in cui appaiono nel vero filesystem principale. Si deve copiare l'intero contenuto dell'installazione LilyPond (ad esempio, '`/usr/share/lilypond`').

  Se c'è un problema, il modo più semplice per individuarlo è lanciare LilyPond usando `strace`, che permetterà di scoprire quali file mancano.

Eseguire LilyPond
: In una gabbia montata con `noexec` è impossibile eseguire qualsiasi programma esterno. Dunque LilyPond deve essere eseguito con un backend che non richieda tale programma. Come è già stato detto, deve essere eseguito con privilegi di superutente (che ovviamente perderà immediatamente), possibilmente con l'uso di `sudo`. È una buona idea limitare il numero di secondi di tempo della CPU che LilyPond può usare (ad esempio con `ulimit -t`), e, se il sistema operativo lo permette, la quantità di memoria che può essere allocata. Si veda anche ⟨undefined⟩ [LilyPond in chroot jail], pagina ⟨undefined⟩.

`-l,--loglevel=`*LIVELLO*
: Imposta la verbosità dell'output della console su *LIVELLO*. I valori possibili sono:

  `NONE` Nessun output, nemmeno i messaggi di errore.

  `ERROR` Solo i messaggi di errore, niente avvisi o messaggi di elaborazione.

  `WARN` Avvisi e nessaggi di errore, nessun messaggio di elaborazione.

  `BASIC_PROGRESS`
  : Messaggi di elaborazione di base (riuscita), avvisi e errori.

  `PROGRESS` Tutti i messaggi di elaborazione, avvisi e errori.

  `INFO (predefinito)`
  : Messaggi di elaborazione, avvisi, errori e ulteriori informazioni di esecuzione.

  `DEBUG` Tutti i messaggi possibili, incluso l'output verboso di debug.

`-o,--output=`*FILE* o *CARTELLA*
: Imposta il file di output predefinito *FILE* oppure, se una cartella con quel nome esiste già, dirige l'output in *CARTELLA*, prendendo il nome del file dal file di

input. In entrambi i casi verrà aggiunto il suffisso appropriato (ad esempio `.pdf` per il pdf).

`--ps` Genera PostScript.

`--png` Genera immmagini di ogni pagina in formato PNG. Questo implica '`--ps`'. La risoluzione in DPI dell'immagine può essere impostata con

```
-dresolution=110
```

`--pdf` Genera PDF. Questo implica '`--ps`'.

`-v,--version`
Mostra informazioni sulla versione.

`-V,--verbose`
Aumenta la prolissità: mostra i percorsi completi di tutti i file letti e dà informazioni sui tempi.

`-w,--warranty`
Mostra la garanzia con cui viene distribuito GNU LilyPond. (Distribuito con **NESSUNA GARANZIA**!)

## Opzioni avanzate della linea di comando per `lilypond`

`-d`*`[nome-opzione]=[valore]`*`,--define-default=`*`[nome-opzione]=[valore]`*
Imposta l'equivalente funzione interna di Scheme su *valore*.

```
-dbackend=svg
```

Se non viene specificato un *valore*, viene usato il valore predefinito. Per disabilitare un'opzione, si può usare il prefisso `no-` prima di *nome-opzione*, ad esempio

```
-dpoint-and-click=#f
```

è equivalente a

```
-dno-point-and-click
```

Sono supportate le seguenti opzioni insieme ai loro rispettivi valori predefiniti:

| **Simbolo** | **Valore** | **Spiegazione/Opzioni** |
|---|---|---|
| `anti-alias-factor` | `1` | Elabora a una risoluzione più alta (usando un certo fattore) e ridimensiona il risultato per evitare gli 'artefatti' nelle immagini `PNG`. |
| `aux-files` | `#t` | Crea i file `.tex`, `.texi`, `.count` nel backend `EPS`. |
| `backend` | `ps` | Seleziona il backend. I file Postscript (predefinito) includono i tipi di carattere `TTF`, `Type1` e `OTF`. Non vengono inclusi i sottoinsiemi di questi tipi. Se si usa un set di caratteri 'orientali', si possono ottenere file di grosse dimensioni. |
| | `eps` | Encapsulated PostScript. Invia ogni pagina (sistema) in un file '`EPS`' separato, senza font, e in un unico file '`EPS`' con tutte le pagine (sistemi) inclusi i font. Questa è la modalità predefinita di `lilypond-book`. |

| | | |
|---|---|---|
| | `null` | non produce la stampa della partitura; ha lo stesso effetto di `-dno-print-pages`. |
| | `svg` | Scalable Vector Graphics. Crea un singolo file SVG, senza font incorporati, per ogni pagina dell'output. Si raccomanda di installare i font Century Schoolbook, inclusi nell'installazione di LilyPond, per una resa ottimale. In UNIX basta copiare questi font dalla directory di LilyPond (solitamente '`/usr/share/lilypond/VERSION/fonts/otf/`') in '`~/.fonts/`'. L'output SVG dovrebbe essere compatibile con qualsiasi editor SVG o user agent. C'è anche un'opzione `svg-woff` (sotto) per poter usare i file di font woff nel backend SVG. |
| | `scm` | Estrae i comandi di disegno grezzi e interni, basati su Scheme. |
| `check-internal-types` | `#f` | Controlla l'assegnazione di ogni proprietà per i tipi. |
| `clip-systems` | `#f` | Genera frammenti ritagliati di una partitura. |
| `datadir` | | Prefisso per i file di dati (sola lettura). |
| `debug-gc` | `#f` | Scarica le statistiche sul debug della memoria. |
| `debug-gc-assert-parsed-dead` | `#f` | Per il debug della memoria: Assicura che tutti i riferimenti agli oggetti analizzati siano eliminati. Questa è un'opzione interna e viene abilitata automaticamente da `` `-ddebug-gc' ``. |
| `debug-lexer` | `#f` | Debug dell'analizzatore lessicale flex. |
| `debug-page-breaking-scoring` | `#f` | Crea le partiture per diverse configurazioni di interruzione di pagina. |
| `debug-parser` | `#f` | Debug dell'analizzatore bison. |
| `debug-property-callbacks` | `#f` | Debug delle catene cicliche di callback. |
| `debug-skylines` | `#f` | Debug skylines. |
| `delete-intermediate-files` | `#t` | Cancella i file `.ps` intermedi e inutilizzabili creati durante la compilazione. |
| `dump-cpu-profile` | `#f` | Scarica l'informazione sui tempi (dipendente dal sistema). |

| | | |
|---|---|---|
| `dump-profile` | `#f` | Scarica l'informazione sulla memoria e il tempo per ogni file. |
| `dump-signatures` | `#f` | Scarica le firme dell'output di ogni sistema. Usato per testare le regressioni. |
| `eps-box-padding` | `#f` | Sposta il margine sinistro della cornice EPS dell'output della quantità specificata (in mm). |
| `gs-load-fonts` | `#f` | Carica i font attraverso Ghostscript. |
| `gs-load-lily-fonts` | `#f` | Carica solo i font LilyPond attraverso Ghostscript. |
| `gui` | `#f` | Esegue il programma senza stampare messaggi e redirige tutto l'output in un file di log. |

**Nota per gli utenti Windows:** Per impostazione predefinita `lilypond.exe` stampa tutta l'informazione sull'avanzamento nella finestra dei comandi. `lilypond-windows.exe` non lo fa e riporta un prompt, privo di informazioni sull'avanzamento, subito nella linea di comando. L'opzione '`-dgui`' può essere usata in questo caso per redirigere l'output in un file di log.

| | | |
|---|---|---|
| `help` | `#f` | Mostra questo aiuto. |
| `include-book-title-preview` | `#t` | Include i titoli dei libri nelle immagini di anteprima. |
| `include-eps-fonts` | `#t` | Include i font in file EPS con sistemi separati. |
| `include-settings` | `#f` | Include il file per le impostazioni globali, questo viene incluso prima che la partitura sia elaborata. |
| `job-count` | `#f` | Elabora in parallelo, usando il dato numero di lavori. |
| `log-file` | `#f [file]` | Se la stringa `FOO` viene assegnata come secondo argomento, redirige l'output nel file `FOO.log`. |
| `max-markup-depth` | `1024` | Maximum depth for the markup tree. If a markup has more levels, assume it will not terminate on its own, print a warning and return a null markup instead. |
| `midi-extension` | `"midi"` | Imposta l'estensione predefinita per il file MIDI sulla stringa specificata. |
| `music-strings-to-paths` | `#f` | Converte le stringhe di testo in percorsi quando i glifi appartengono a un font musicale. |
| `paper-size` | `\"a4\"` | Imposta la dimensione predefinita del foglio. Nota che la stringa deve essere compresa tra virgolette precedute dal segno di escape. |

| | | |
|---|---|---|
| `pixmap-format` | `png16m` | Imposta il formato di output di GhostScript per le immagini raster. |
| `point-and-click` | `#f` | Aggiunge i collegamenti 'punta e clicca' all'output `PDF`. Si veda ⟨undefined⟩ [Point and click], pagina ⟨undefined⟩. |
| `preview` | `#f` | Crea immagini di anteprima oltre al normale output. |

Questa opzione è supportata da tutti i backend; `pdf`, `png`, `ps`, `eps` e `svg`, ma non `scm`. Genera un file di output nella forma `mioFile.preview.estensione`, contenente i titoli e il primo sistema. Se vengono usati i blocchi `\book` o `\bookpart`, i titoli di `\book`, `\bookpart` o `\score` appariranno nell'output, incluso il primo sistema di ogni blocco `\score` se la variabile `print-all-headers` di `\paper` è impistata su `#t`.

Per impedire il normale output, si usano le opzioni '`-dprint-pages`' o '`-dno-print-pages`' in base alle proprie esigenze.

| | | |
|---|---|---|
| `print-pages` | `#t` | Genera le pagine complete (predefinito). '`-dno-print-pages`' è utile in combinazione con '`-dpreview`'. |
| `profile-property-accesses` | `#f` | Mantiene una statistica delle chiamate di funzione `get_property()`. |
| `protected-scheme-parsing` | `#t` | Continua se l'analizzatore coglie degli errori nel codice scheme interno al file di input. Se impostato su `#f`, in caso di errore si ferma e mostra la traccia di stack. |
| `read-file-list` | `#f [file]` | Specifica il nome di un file che contiene una lista di file di input da elaborare. |
| `relative-includes` | `#f` | Quando elabora un comando `\include`, cerca il file incluso in posizione relativa al file corrente (invece che in posizione assoluta). |
| `resolution` | `101` | Imposta la risoluzione per generare immagini `PNG` su un certo valore (in dpi). |
| `safe` | `#f` | Non si fida dell'input nel file `.ly`. |

Quando la formattazione di LilyPond viene messa a disposizione tramite un server web, si **DEVE** passare l'opzione '`--safe`' o l'opzione '`--jail`'. L'opzione '`--safe`' impedirà che il codice Scheme presente nell'input possa fare uno scempio, ad esempio

```
#(system "rm -rf /")
{
  c4^$(ly:gulp-file "/etc/passwd")
}
```

L'opzione '`-dsafe`' serve a valutare le espressioni Scheme presenti nell'input in uno speciale modulo di sicurezza. Questo modulo di sicurezza è derivato dal modulo GUILE '`safe-r5rs`', ma aggiunge alcune funzioni del LilyPond API. Queste funzioni sono elencate in '`scm/safe-lily.scm`'.

Inoltre, la modalità sicura non permette le direttive `\include` e disabilita l'uso del backslash nelle stringhe TEX. In modalità sicura, non è possibile importare le variabili di LilyPond in Scheme.

'`-dsafe`' *non* rileva il sovrautilizzo di risorse. È ancora possibile far sì che il programma rimanga in sospeso per un tempo indefinito, ad esempio alimentando il backend con strutture di dati cicliche. Dunque se si vuole usare LilyPond su un server web pubblicamente accessibile, si deve limitare il processo nell'uso della CPU e della memoria.

La modalità sicura bloccherà la compilazione di molti utili frammenti di codice LilyPond.

L'opzione '`--jail`' è un'alternativa più sicura, ma richiede più lavoro per configurarla. Si veda ⟨undefined⟩ [Basic command line options for LilyPond], pagina ⟨undefined⟩.

| | | |
|---|---|---|
| `separate-log-files` | `#f` | Per i file di input `FILE1.ly`, `FILE2.ly`, etc. salva i dati di log nei file `FILE1.log`, `FILE2.log`, ... |
| `show-available-fonts` | `#f` | Elenca i nomi di font disponibili. |
| `strict-infinity-checking` | `#f` | Forza il blocco del programma quando si incontrano eccezioni `Inf` e `NaN` sui numeri in virgola mobile. |
| `strip-output-dir` | `#t` | Non usa le directory dei file di input per costruire i nomi dei file di output. |
| `svg-woff` | `#f` | Usa i file di font woff nel backend SVG. |
| `trace-memory-frequency` | `#f` | Registra molte volte al secondo l'uso delle celle da parte di Scheme. Salva i risultati in `FILE.stacks` e `FILE.graph`. |
| `trace-scheme-coverage` | `#f` | Registra la copertura dei file Scheme in `FILE.cov`. |
| `verbose` | `#f` | Output dettagliato, ovvero livello di log DEBUG (sola lettura). |
| `warning-as-error` | `#f` | Trasforma tutti i messaggi di avviso e di 'errore di programmazione' in errori. |

## Variabili d'ambiente

`lilypond` riconosce le seguenti variabili d'ambiente:

`LILYPOND_DATADIR`
: Specifica la directory predefinita in cui saranno cercati i messaggi della localizzazione e i file di dati. Questa directory deve contenere sottodirectory chiamate '`ly/`', '`ps/`', '`tex/`', etc.

`LANG`
: Determina la lingua per i messaggi di avviso.

`LILYPOND_LOGLEVEL`
: Il livello di log (loglevel) predefinito. Se LilyPond viene chiamato senza un livello di log esplicito (ovvero senza l'opzione '`--loglevel`' della linea di comando), viene usato questo valore.

LILYPOND_GC_YIELD
: Una variabile, in forma di percentuale, che regola il modo in cui viene gestita la memoria. Con valori più alti il programma usa più memoria, con valori più bassi usa più tempo della CPU. Il valore predefinito è 70.

## LilyPond in una gabbia chroot

Configurare un server perché esegua LilyPond in una gabbia chroot è un lavoro complesso. La procedura è spiegata sotto. Gli esempi si riferiscono a Ubuntu GNU/Linux e potrebbero richiedere l'uso di sudo in alcune situazioni.

- Installa i pacchetti necessari: LilyPond, GhostScript e ImageMagick.
- Crea un nuovo utente dal nome lily:

  ```
  adduser lily
  ```

  Questo comando creerà anche un nuovo gruppo per l'utente lily, e una cartella home, /home/lily
- Nella cartella home dell'utente lily crea un file da usare come filesystem separato:

  ```
  dd if=/dev/zero of=/home/lily/loopfile bs=1k count= 200000
  ```

  In questo esempio è stato creato un file di 200MB da usare come filesystem della gabbia.
- Crea un dispositivo di loop, crea e monta un filesystem, quindi crea una cartella scrivibile dall'utente lily:

  ```
  mkdir /mnt/lilyloop
  losetup /dev/loop0 /home/lily/loopfile
  mkfs -t ext3 /dev/loop0 200000
  mount -t ext3 /dev/loop0 /mnt/lilyloop
  mkdir /mnt/lilyloop/lilyhome
  chown lily /mnt/lilyloop/lilyhome
  ```
- Nella configurazione dei server, JAIL sarà /mnt/lilyloop e DIR sarà /lilyhome.
- Crea un grande albero delle directory nella gabbia copiando i file necessari, come mostrato nello script di esempio più in basso.

  Puoi usare sed per creare i comandi di copia necessari per un certo eseguibile:

  ```
  for i in "/usr/local/lilypond/usr/bin/lilypond" "/bin/sh" "/usr/bin/;  \
    do ldd $i | sed 's/.*=> \/\(.*\/\)\([^(]*\).*/mkdir -p \1 \&\&  \
      cp -L \/\1\2 \1\2/' | sed 's/\t\/\(.*\/\)\(.*\) (.*)$/mkdir -p  \
        \1 \&\& cp -L \/\1\2 \1\2/' | sed '/.*=>.*/d'; done
  ```

## Script di esempio per Ubuntu 8.04 a 32-bit

```
#!/bin/sh
## defaults set here

username=lily
home=/home
loopdevice=/dev/loop0
jaildir=/mnt/lilyloop
# the prefix (without the leading slash!)
lilyprefix=usr/local
# the directory where lilypond is installed on the system
lilydir=/$lilyprefix/lilypond/

userhome=$home/$username
loopfile=$userhome/loopfile
```

```
adduser $username
dd if=/dev/zero of=$loopfile bs=1k count=200000
mkdir $jaildir
losetup $loopdevice $loopfile
mkfs -t ext3 $loopdevice 200000
mount -t ext3 $loopdevice $jaildir
mkdir $jaildir/lilyhome
chown $username $jaildir/lilyhome
cd $jaildir

mkdir -p bin usr/bin usr/share usr/lib usr/share/fonts $lilyprefix tmp
chmod a+w tmp

cp -r -L $lilydir $lilyprefix
cp -L /bin/sh /bin/rm bin
cp -L /usr/bin/convert /usr/bin/gs usr/bin
cp -L /usr/share/fonts/truetype usr/share/fonts

# Now the library copying magic
for i in "$lilydir/usr/bin/lilypond" "$lilydir/usr/bin/guile" "/bin/sh"  \
  "/bin/rm" "/usr/bin/gs" "/usr/bin/convert"; do ldd $i | sed 's/.*=>  \
    \/\(.*\/\)\([^(]*\).*/mkdir -p \1 \&\& cp -L \/\1\2 \1\2/' | sed  \
      's/\t\/\(.*\/\)\(.*\) (.*)$/mkdir -p \1 \&\& cp -L \/\1\2 \1\2/'  \
        | sed '/.*=>.*/d'; done | sh -s

# The shared files for ghostscript...
      cp -L -r /usr/share/ghostscript usr/share
# The shared files for ImageMagick
      cp -L -r /usr/lib/ImageMagick* usr/lib

### Now, assuming that you have test.ly in /mnt/lilyloop/lilyhome,
### you should be able to run:
### Note that /$lilyprefix/bin/lilypond is a script, which sets the
### LD_LIBRARY_PATH - this is crucial
      /$lilyprefix/bin/lilypond -jlily,lily,/mnt/lilyloop,/lilyhome test.ly
```

## 1.3 Messaggi di errore

Quando si compila un file possono apparire vari messaggi di errore:

*Avvertimento*
: Qualcosa appare sospetto. Se stai cercando di fare qualcosa di insolito allora comprenderai il messaggio e potrai ignorarlo. Tuttavia di solito i messaggi di avvertimento indicano che il file di input ha qualcosa che non va.

*Errore*
: C'è qualcosa di assolutamente sbagliato. Il passo attualmente in elaborazione (analisi, interpretazione o formattazione) verrà completato, ma il passo successivo verrà saltato.

*Errore fatale*
: C'è qualcosa di assolutamente sbagliato e LilyPond non può continuare. Questo accade raramente. La causa più comune è un'errata installazione dei tipi di carattere.

*Errore Scheme*
: Gli errori che capitano mentre si esegue del codice Scheme sono individuati dall'interprete Scheme. Se si esegue con l'opzione di prolissità (`-V` o '`--verbose`'), viene stampata una traccia della chiamata di funzione responsabile dell'errore.

*Errore di programmazione*
: Si è verificata una qualche incongruenza interna. Questi messaggi di errore servono ad aiutare programmatori e debugger. Di solito si possono ignorare. Talvolta sono talmente numerosi da nascondere il resto dell'output.

*Sospeso (core dumped)*
: Segnala un serio errore di programmazione che ha mandato in crash il programma. Questi errori sono considerati critici. Se ti imbatti in un errore simile, invia una segnalazione di errore.

Se gli avvertimenti e gli errori possono essere collegati a una parte specifica del file di input, i messaggi di errore hanno la seguente forma

```
file:riga:colonna: messaggio
riga di input responsabile dell'errore
```

Nella riga responsabile si inserisce un a capo per indicare la colonna in cui è stato trovato l'errore. Ad esempio,

```
test.ly:2:19: error: not a duration: 5
  { c'4 e'
          5 g' }
```

Queste posizioni indicano la migliore ipotesi di LilyPond a proposito del punto in cui l'avvertimento o l'errore sono comparsi, ma (per loro stessa natura) avvertimenti ed errori capitano quando succede qualcosa di imprevisto. Se non riesci a vedere un errore nella riga indicata del file di input, prova a controllare una o due righe sopra la posizione indicata.

Maggiori informazioni sugli errori si trovano in Sezione 1.4 [Errori comuni], pagina 11.

## 1.4 Errori comuni

Le condizioni di errore descritte di seguito capitano spesso, ma la causa non è ovvia né facile da trovare. Una volta che sono state individuate e comprese, è facile gestirle.

### La musica esce dalla pagina

Se la musica esce dalla pagina al di là del margine destro o appare eccessivamente compressa, quasi sempre è dovuto all'inserimento di una durata errata di una nota, che fa sì che l'ultima nota di una misura si estenda oltre la barra di divisione. Non è sbagliato se la nota finale di una misura non termina entro la barra di divisione inserita automaticamente, perché semplicemente si assume che la nota continui nella misura successiva. Ma se si presenta una lunga sequenza di misure simili, la musica può apparire compressa o può uscire dalla pagina perché gli a capo automatici possono essere inseriti soltanto alla fine di misure complete, ovvero quando tutte le note finiscono prima o alla fine della misura.

> **Nota:** Una durata sbagliata può inibire l'interruzione di linea, portando a una linea di musica estremamente compressa o a musica che esce dalla pagina.

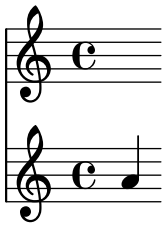
La durata errata può essere trovata facilmente se si usano i controlli di battuta, si veda Sezione "Controlli di battuta e del numero di battuta" in *Guida alla Notazione*.

Se si vuole davvero ottenere una serie di tali misure sovrapposte bisogna inserire una barra di divisione invisibile nel punto in cui si desidera l'interruzione di linea. Per i dettagli si veda Sezione "Stanghette" in *Guida alla Notazione*.

### Appare un rigo in più

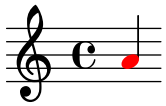
Se i contesti non sono creati esplicitamente con `\new` o `\context`, saranno creati senza avviso appena si incontra un comando che non può essere applicato a un contesto esistente. Nelle partiture semplici la creazione automatica dei contesti è utile: infatti la maggior parte degli esempi nei manuali LilyPond sfrutta questa semplificazione. Talvolta, però, la creazione silenziosa di contesti può causare la comparsa di nuovi righi o partiture non desiderate. Ad esempio, si potrebbe pensare che il seguente codice colori di rosso tutte le teste delle note nel rigo, ma in realtà produce due righi, di cui il più basso conserva il colore nero predefinito per le teste delle note.

```
\override Staff.NoteHead.color = #red
\new Staff { a }
```



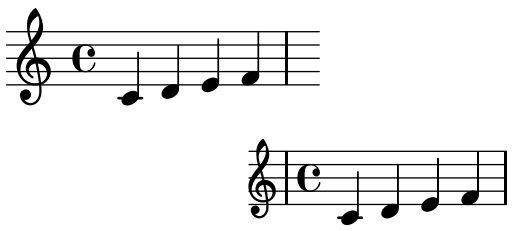
Questo accade perché non esiste un contesto `Staff` quando viene elaborata l'istruzione di override, quindi ne viene implicitamente creato uno e l'override viene applicato ad esso. Ma poi il comando `\new Staff` crea un altro rigo separato nel quale vengono inserite le note. Il codice corretto per colorare le teste di tutte le note è

```
\new Staff {
  \override Staff.NoteHead.color = #red
  a
}
```



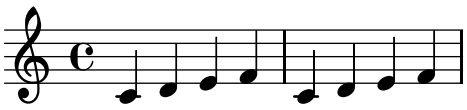
Vediamo un secondo esempio. Se un comando `\relative` viene posto dentro un comando `\repeat`, vengono generati due righi, il secondo spostato orizzontalmente rispetto al primo, perché il comando `\repeat` genera due blocchi `\relative`, ognuno dei quali crea implicitamente i blocchi `Staff` e `Voice`.

```
\repeat unfold 2 {
  \relative c' { c4 d e f }
}
```



Per correggere il problema basta istanziare esplicitamente il contesto `Voice`:

```
\new Voice {
  \repeat unfold 2 {
    \relative c' { c4 d e f }
  }
}
```



### Messaggio di errore Unbound variable %

Questo messaggio di errore comparirà in fondo alla console di output o nel file di log insieme al messaggio "GUILE signalled an error . . ." ogni volta che viene chiamata una routine di Scheme che contenga (erroneamente) un commento *LilyPond* invece di un commento *Scheme*.

I commenti LilyPond iniziano con un segno di percentuale, (`%`), e non devono essere usati all'interno delle routine di Scheme. I commenti Scheme iniziano con un punto e virgola, (`;`).

### Messaggio di errore FT_Get_Glyph_Name

Questo messaggio di errore compare nella console di output o nel file di log file se un file di input contiene un carattere non-ASCII e non è stato salvato nella codifica UTF-8. Per dettagli si veda Sezione "Text encoding" in *Guida alla Notazione*.

### Avvertimento sul fatto che le affinità del rigo devono solo diminuire

Questo avvertimento può apparire se non ci sono dei righi nell'output, ad esempio se ci sono solo un contesto `ChordName` e un contesto `Lyrics`, come in un lead sheet. Si possono evitare questi messaggi di avvertimento facendo in modo che uno dei contesti si comporti come un rigo inserendo

```
\override VerticalAxisGroup.staff-affinity = ##f
```

all'inizio del contesto. Per dettagli si veda "Spacing of non-staff lines" in Sezione "Flexible vertical spacing within systems" in *Guida alla Notazione*.

### Messaggio di errore \new inaspettato

Un blocco `\score` deve contenere una *singola* espressione musicale. Se invece contiene vari `\new Staff`, `\new StaffGroup` o simili contesti introdotti con `\new` senza che questi siano racchiusi tra parentesi graffe, `{ ... }`, o doppie parentesi uncinate, `<< ... >>`, ovvero così:

```
\score {
  % Invalido! Genera l'errore: errore di sintassi, \new inaspettato
  \new Staff { ... }
  \new Staff { ... }
}
```

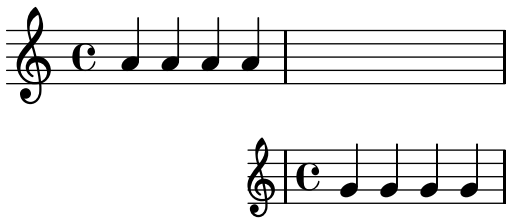
verrà generato questo messaggio di errore.

Per evitare l'errore, è sufficiente racchiudere tutti i blocchi `\new` tra parentesi graffe o doppie parentesi uncinate.

Se si usano le parentesi graffe, i blocchi `\new` appariranno in modo sequenziale:

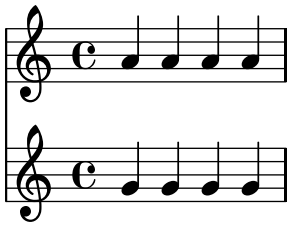
```
\score {
  {
    \new Staff { a' a' a' a' }
    \new Staff { g' g' g' g' }
  }
```

```
}
```



ma è più probabile che si debbano usare le doppie parentesi uncinate in modo che i nuovi righi siano avviati in parallelo, ovvero contemporaneamente:

```
\score {
  <<
    \new Staff { a' a' a' a' }
    \new Staff { g' g' g' g' }
  >>
}
```



# 2 Aggiornare i file con convert-ly

La sintassi di input di LilyPond viene regolarmente modificata per semplificarla o per migliorarla in vari modi. L'effetto collaterale è che l'interprete di LilyPond spesso non è più compatibile con i vecchi file di input. Per ovviare a questo problema, si può usare il programma `convert-ly`, che permette di gestire gran parte dei cambiamenti di sintassi tra le versioni di LilyPond.

## 2.1 Perché la sintassi cambia?

La sintassi di input di LilyPond talvolta cambia. Via via che LilyPond migliora, la sintassi (il linguaggio dell'input) viene modificata di conseguenza. Queste modifiche vengono fatte a volte per far sì che l'input sia più facile da leggere e da scrivere e a volte per aggiungere a LilyPond nuove funzionalità.

Ad esempio, tutti i nomi delle proprietà di `\paper` e `\layout` dovrebbero essere scritte nella forma `primo-secondo-terzo`. Tuttavia, nella versione 2.11.60 ci siamo accorti che la proprietà `printallheaders` non seguiva questa convenzione. Dovevamo lasciarla così come era (confondendo i nuovi utenti che devono avere a che fare con un formato di input incoerente), o cambiarla (disturbando i vecchi utenti che avevano già delle partiture)? In questo caso decidemmo di cambiare il nome in `print-all-headers`. Fortunatamente, questa modifica può essere automatizzata con `convert-ly`.

Purtroppo `convert-ly` non è in grado di gestire tutti i cambiamenti dell'input. Ad esempio, in LilyPond 2.4 e precedenti, gli accenti e le lettere non inglesi venivano inserite con LaTeX – per mostrare la parola francese per Natale si usava `No\"el`. Ma in LilyPond 2.6 e superiori, il carattere speciale ë deve essere inserito direttamente nel file LilyPond come carattere UTF-8. `convert-ly` non può sostituire tutti i caratteri speciali di LaTeX con i rispettivi caratteri UTF-8; è necessario aggiornare a mano i vecchi file di input di LilyPond.

## 2.2 Utilizzo di convert-ly

`convert-ly` usa la dichiarazione `\version` nel file di input per determinare il vecchio numero di versione. Nella maggior parte dei casi per aggiornare il file di input è sufficiente eseguire

```
convert-ly -e miofile.ly
```

nella directory che contiene il file. Questo comando aggiornerà '`miofile.ly`' e preserverà il file originale in '`miofile.ly~`'.

> **Nota:** `convert-ly` converte sempre fino all'ultimo cambiamento di sintassi gestito. Questo significa che il numero di `\version` che appare nel file convertito è di solito inferiore al numero di versione di `convert-ly`.

Per convertire in una volta sola tutti i file di input in una directory si usa

```
convert-ly -e *.ly
```

Altrimenti, se si desidera specificare un nome diverso per il file aggiornato, senza modificare il file originale e il suo nome, si usa

```
convert-ly miofile.ly > mionuovofile.ly
```

Il programma elencherà i numeri di versione per i quali sono state eseguite le conversioni. Se non vengono elencati dei numeri di versione il file è già aggiornato.

Gli utenti MacOS X possono eseguire questi comandi dalla voce di menu `Compila > Aggiorna la sintassi`.

Gli utenti Windows devono inserire questi comandi nella finestra del Prompt dei comandi, che di solito si trova in `Start > Accessori > Prompt dei comandi`.

## 2.3 Opzioni da linea di comando per convert-ly

Il programma viene lanciato in questo modo:

```
convert-ly [opzione]... nomefile...
```

Esistono le seguenti opzioni:

`-d,--diff-version-update`
: aggiorna `\version` all'ultima versione oppure, se questa è maggiore, non fa niente.

`-e,--edit`
: Applica le conversioni direttamente nel file di input, modificando l'originale.

`-f,--from=`*from-patchlevel*
: Imposta la versione da cui convertire. Se non viene impostata, `convert-ly` la ricaverà dalla stringa `\version` presente nel file. Esempio: '`--from=2.10.25`'

`-h, --help`
: Mostra la schermata di aiuto.

`-l` *loglevel*`, --loglevel=`*loglevel*
: Imposta la verbosità dell'output su *loglevel*. I valori possibili, in caratteri maiuscoli, sono `PROGRESS` (predefinito), `NONE`, `WARNING`, `ERROR` e `DEBUG`.

`-n,--no-version`
: Normalmente `convert-ly` aggiunge un indicatore `\version` nell'output. Questa opzione lo impedisce.

`-s, --show-rules`
: Mostra tutte le conversioni conosciute ed esce.

`-t, --to=`*to-patchlevel*
: Imposta esplicitamente la versione obiettivo della conversione, altrimenti viene usato il valore più recente.

```
convert-ly --to=2.14.1 miofile.ly
```

Per aggiornare i frammenti LilyPond presenti nei file texinfo, si usa

```
convert-ly --from=... --to=... --no-version *.itely
```

Per vedere i cambiamenti della sintassi di LilyPond tra due versioni, si usa

```
convert-ly --from=... --to=... -s
```

## 2.4 Problemi nell'eseguire convert-ly

Quando si esegue convert-ly in una finestra del Prompt dei comandi in Windows su un file il cui nome o percorso contengano degli spazi, è necessario includere tutto il nome del file di input con tre (!) virgolette doppie:

```
convert-ly """D:/Mie Partiture/Ode.ly""" > "D:/Mie Partiture/new Ode.ly"
```

Se il semplice comando `convert-ly -e *.ly` non funziona perché la linea di comando espansa diventa troppo lunga, si può inserire il comando `convert-ly` in un loop. Questo esempio per UNIX aggiornerà tutti i file '`.ly`' nella directory corrente

```
for f in *.ly; do convert-ly -e $f; done;
```

Nella finestra del Prompt dei comandi di Windows il comando corrispondente è

```
for %x in (*.ly) do convert-ly -e """%x"""
```

Non vengono gestiti tutti i cambiamenti del linguaggio. Si può specificare solo un'opzione di output. È piuttosto improbabile che si aggiornino automaticamente il codice scheme e le interfacce di scheme di LilyPond; tieniti pronto a correggere a mano il codice scheme.

## 2.5 Conversioni manuali

In teoria, un programma come convert-ly potrebbe gestire qualsiasi cambiamento di sintassi. Dopo tutto, un programma per computer interpreta la vecchia versione e la nuova versione, quindi un altro programma può tradurre un file in un altro[1].

Tuttavia il progetto LilyPond ha risorse limitate: non tutte le conversioni sono compiute automaticamente. Di seguito è riportato l'elenco dei problemi noti.

```
1.6->2.0:
 Doesn't always convert figured bass correctly, specifically things like {<
>}.  Mats' comment on working around this:
   To be able to run convert-ly
   on it, I first replaced all occurrences of '{<' to some dummy like '{#'
   and similarly I replaced '>}' with '&}'.  After the conversion, I could
   then change back from '{ #' to '{ <' and from '& }' to '> }'.
 Doesn't convert all text markup correctly.  In the old markup syntax,
 it was possible to group a number of markup commands together within
parentheses, e.g.
   -#'((bold italic) "string")
   This will incorrectly be converted into
   -\markup{{\bold italic} "string"}
   instead of the correct
   -\markup{\bold \italic "string"}
2.0->2.2:
 Doesn't handle \partcombine
 Doesn't do \addlyrics => \lyricsto, this breaks some scores with multiple
stanzas.
2.0->2.4:
 \magnify isn't changed to \fontsize.
    - \magnify #m => \fontsize #f, where f = 6ln(m)/ln(2)
 remove-tag isn't changed.
    - \applyMusic #(remove-tag '. . .) => \keepWithTag #'. . .
 first-page-number isn't changed.
    - first-page-number no => print-first-page-number = ##f
 Line breaks in header strings aren't converted.
    - \\\\  as line break in \header strings => \markup \center-align <
      "First Line" "Second Line" >
 Crescendo and decrescendo terminators aren't converted.
    - \rced => \!
    - \rc => \!
2.2->2.4:
 \turnOff (used in \set Staff.VoltaBracket = \turnOff) is not properly
converted.
2.4.2->2.5.9
 \markup{ \center-align <{ ... }> } should be converted to:
 \markup{ \center-align {\line { ... }} }
 but now, \line is missing.
2.4->2.6
 Special LaTeX characters such as $~$ in text are not converted to UTF8.
2.8
```

[1] O almeno questo è possibile in qualsiasi file LilyPond che non contenga codice scheme. Se c'è del codice scheme nel file, allora il file LilyPond contiene un linguaggio Turing-completo, ed è possibile imbattersi in problemi col famigerato "Problema dell'arresto" in informatica.

```
\score{} must now begin with a music expression.  Anything else
(particularly \header{}) must come after the music.
```

# 3 Eseguire lilypond-book

Se si desidera aggiungere a un documento illustrazioni musicali, si può semplicemente fare nello stesso modo in cui si farebbe con altri tipi di immagini: prima si creano le immagini separatamente, in formato PostScript o PNG, poi le si includono in un documento LaTeX o HTML.

`lilypond-book` offre la possibilità di automatizzare tale procedimento: questo programma estrae i frammenti musicali dal documento, esegue `lilypond` su di essi e crea un nuovo documento contenente le illustrazioni musicali così ottenute. Le definizioni relative alla larghezza del rigo e alle dimensioni dei caratteri vengono regolate per adeguarsi alla formattazione del documento.

Si tratta di un programma separato da `lilypond` e viene lanciato dalla linea di comando; per maggiori informazioni, si veda ⟨undefined⟩ [Command-line usage], pagina ⟨undefined⟩. In caso di problemi nell'eseguire `lilypond-book` da linea di comando su Windows o Mac OS X, si veda Sezione "Windows" in *Informazioni generali* o Sezione "MacOS X" in *Informazioni generali*.

Questo procedimento può essere applicato ai documenti LaTeX, HTML, Texinfo o DocBook.

## 3.1 Un esempio di documento musicologico

Alcuni testi contengono degli esempi musicali: si tratta di trattati musicologici, canzonieri o manuali come questo. È possibile crearli a mano, semplicemente importando un'immagine PostScript nell'elaboratore di testo. Tuttavia esiste una procedura automatizzata che permette di ridurre il carico di lavoro richiesto dai documenti in formato HTML, LaTeX, Texinfo e DocBook.

Uno script chiamato `lilypond-book` estrarrà i frammenti musicali, li formatterà e restituirà la notazione risultante. Ecco un piccolo esempio da usare con LaTeX. L'esempio contiene anche del testo esplicativo, dunque non è necessario entrare nei dettagli.

### Input

```
\documentclass[a4paper]{article}

\begin{document}

I documenti per \verb+lilypond-book+ possono combinare liberamente musica e testo.
Ad esempio,

\begin{lilypond}
\relative c' {
  c2 e2 \tuplet 3/2 { f8 a b } a2 e4
}
\end{lilypond}

Le opzioni vengono specificate tra parentesi quadre.

\begin{lilypond}[fragment,quote,staffsize=26,verbatim]
  c'4 f16
\end{lilypond}

Se l'esempio è più grande, è possibile metterlo in un file separato e inserirlo
con \verb+\lilypondfile+.

\lilypondfile[quote,noindent]{screech-and-boink.ly}
```

```
(Se vuoi provare, sostituisci @file{screech-and-boink.ly} con qualsiasi file @file{.ly
che si trovi nella stessa directory di questo file.)

\end{document}
```

## Elaborazione

Salva il codice precedente in un file chiamato '`lilybook.lytex`', quindi esegui in un terminale

```
lilypond-book --output=out --pdf lilybook.lytex
lilypond-book (GNU LilyPond) 2.17.26

Lettura di lilybook.lytex...
...tagliato molto output...
Compilazione di lilybook.tex...
cd out
pdflatex lilybook.tex
...tagliato molto output...
xpdf lilybook.pdf
(sostituisci xpdf col tuo lettore PDF preferito)
```

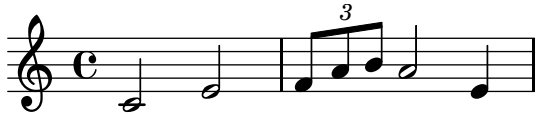
L'esecuzione di `lilypond-book` e `pdflatex` crea molti file temporanei e questo potrebbe rendere disordinata la directory di lavoro. Per ovviare a questo inconveniente, è consigliabile usare l'opzione `--output=dir`. In questo modo i file verranno salvati in una sottodirectory '`dir`' separata.

Infine ecco il risultato dell'esempio in LaTeX.[1] Si conclude qui la parte di tutorial.

[1] Questo tutorial è elaborato da Texinfo, dunque l'esempio produce dei risultati leggermente diversi nella formattazione.

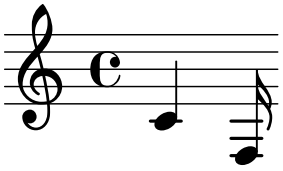
## Output

I documenti per \verb+lilypond-book+ possono combinare liberamente musica e testo. Ad esempio,

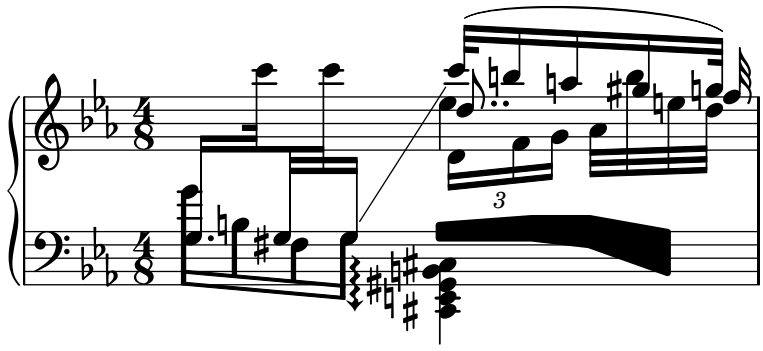


Le opzioni vengono specificate tra parentesi quadre.

```
c'4 f16
```



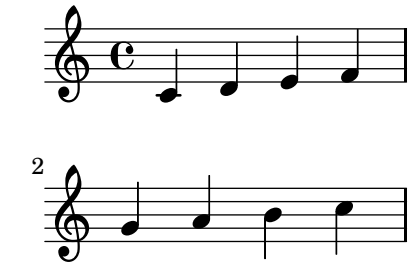
Se l'esempio è più grande, è possibile riportarlo in un file a parte e inserirlo con \verb+\lilypondfile+.



Perché sia visibile la `tagline`, predefinita o personalizzata, l'intero frammento deve essere compreso in un costrutto `\book { }`.

```
\book{
  \header{
    title = "Una scala in LilyPond"
  }

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

# na scala in LilyP




aving by LilyPond 2.17.26—wv

## 3.2 Integrare musica e testo

Questa sezione spiega come integrare LilyPond in vari formati di output.

### 3.2.1 LaTeX

LaTeX costituisce lo standard de facto per le pubblicazioni nell'ambito delle scienze esatte. Si basa sul motore tipografico TeX, che produce la migliore qualità tipografica possibile.

Si veda *The Not So Short Introduction to LaTeX* per una panoramica sull'uso di LaTeX.

`lilypond-book` fornisce i seguenti comandi e ambienti per includere la musica nei file LaTeX:

- il comando `\lilypond{...}`, dove si può inserire direttamente del codice lilypond corto
- l'ambiente `\begin{lilypond}...\end{lilypond}`, dove si può inserire direttamente del codice lilypond più lungo
- il comando `\lilypondfile{...}` per inserire un file lilypond
- il comando `\musicxmlfile{...}` per inserire un file MusicXML, che sarà elaborato da `musicxml2ly` e da `lilypond`.

Nel file di input, la musica viene specificata con uno dei seguenti comandi:

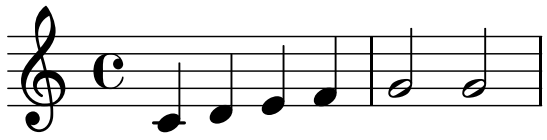
```
\begin{lilypond}[le,opzioni,vanno,qui]
  CODICE LILYPOND
\end{lilypond}

\lilypond[le,opzioni,vanno,qui]{ CODICE LILYPOND }

\lilypondfile[le,opzioni,vanno,qui]{nomefile}

\musicxmlfile[le,opzioni,vanno,qui]{nomefile}
```

Inoltre, `\lilypondversion` mostra la versione di lilypond impiegata. L'esecuzione di `lilypond-book` produce un file che può essere ulteriormente elaborato con LaTeX.

Vediamo alcuni esempi. L'ambiente `lilypond`

```
\begin{lilypond}[quote,fragment,staffsize=26]
  c' d' e' f' g'2 g'2
\end{lilypond}
```

genera



La versione breve

```
\lilypond[quote,fragment,staffsize=11]{<c' e' g'>}
```

genera



Attualmente non è possibile includere `{` o `}` all'interno di `\lilypond{}`, dunque questo comando è utile solo se usato con l'opzione `fragment`.

La lunghezza predefinita del rigo musicale sarà regolata in base ai comandi presenti nel preambolo del documento, ovvero la parte del documento che precede `\begin{document}`. Il

comando `lilypond-book` li invia a LaTeX per definire la larghezza del testo. La larghezza del rigo nei frammenti musicali è quindi regolato in base alla larghezza del testo. Si noti che questo algoritmo euristico può fallire facilmente; in questi casi occorre usare l'opzione `line-width` nel frammento musicale.

Ogni frammento chiamerà le seguenti macro se sono state definite dall'utente:

- `\preLilyPondExample` prima della musica,
- `\postLilyPondExample` dopo la musica,
- `\betweenLilyPondSystem[1]` tra i sistemi, se `lilypond-book` ha diviso il frammento in più file PostScript. Prende un parametro e riceve tutti i file già inclusi in questo frammento. Per impostazione predefinita inserisce semplicemente un `\linebreak`.

## Frammenti di codice selezionati

Talvolta si ha necessità di mostrare degli elementi musicali (ad esempio le legature di portamento e di valore) che proseguono dopo la fine del frammento. È possibile inserirli mandando il rigo a capo e impedendo l'inclusione del restante output di LilyPond.

In LaTeX, si definisce `\betweenLilyPondSystem` in modo tale che l'inclusione di altri sistemi venga terminata una volta incluso il numero di sistemi richiesti. Dato che `\betweenLilyPondSystem` viene chiamato la prima volta *dopo* il primo sistema, includere solo il primo sistema è semplice.

```
\def\betweenLilyPondSystem#1{\endinput}

\begin{lilypond}[fragment]
  c'1\( e'( c'~ \break c' d) e f\)
\end{lilypond}
```

Se serve una maggior quantità di sistemi, occorre usare un condizionale TeX prima di `\endinput`. In questo esempio, sostituisci '2' col numero di sistemi che desideri avere nell'output.

```
\def\betweenLilyPondSystem#1{
    \ifnum#1<2\else\expandafter\endinput\fi
}
```

(Dato che `\endinput` arresta immediatamente l'elaborazione del file di input corrente, occorre usare `\expandafter` per ritardare la chiamata di `\endinput` e far sì che avvenga dopo l'esecuzione di `\fi`, in modo da bilanciare la condizione `\if-\fi`.)

Ricorda che la definizione di `\betweenLilyPondSystem` è efficace finché TeX esce dal gruppo attuale (ad esempio l'ambiente LaTeX) o è sovrascritto da un'altra definizione (che vale, in gran parte dei casi, per il resto del documento). Per reimpostare la definizione, si scrive

```
\let\betweenLilyPondSystem\undefined
```

nel sorgente LaTeX.

Si potrebbe semplificare la procedura creando una macro TeX

```
\def\onlyFirstNSystems#1{
    \def\betweenLilyPondSystem##1{%
      \ifnum##1<#1\else\expandafter\endinput\fi}
}
```

e poi specificando prima di ogni frammento la quantità di sistemi desiderata,

```
\onlyFirstNSystems{3}
\begin{lilypond}...\end{lilypond}
\onlyFirstNSystems{1}
\begin{lilypond}...\end{lilypond}
```

### Vedi anche

Esistono opzioni specifiche da linea di comando per `lilypond-book` e altri dettagli da conoscere quando si elaborano documenti LaTeX; si veda ⟨undefined⟩ [Invoking lilypond-book], pagina ⟨undefined⟩.

## 3.2.2 Texinfo

Texinfo è il formato standard per la documentazione del progetto GNU. Un esempio di documento Texinfo è questo stesso manuale. Le versioni del manuale in formato HTML, PDF e Info vengono generate da un documento Texinfo.

`lilypond-book` fornisce i seguenti comandi e ambienti per includere musica nei file Texinfo:

- il comando `\lilypond{...}`, dove si può inserire direttamente del codice lilypond corto
- l'ambiente `\begin{lilypond}...\end{lilypond}`, dove si può inserire direttamente del codice lilypond più lungo
- il comando `\lilypondfile{...}` per inserire un file lilypond
- il comando `\musicxmlfile{...}` per inserire un file MusicXML, che sarà elaborato da `musicxml2ly` e da `lilypond`.

Nel file di input, la musica viene specificata con uno dei seguenti comandi

```
@lilypond[le,opzioni,vanno,qui]
  IL TUO CODICE LILYPOND
@end lilypond

@lilypond[le,opzioni,vanno,qui]{ IL TUO CODICE LILYPOND }

@lilypondfile[le,opzioni,vanno,qui]{nomefile}

@musicxmlfile[le,opzioni,vanno,qui]{nomefile}
```

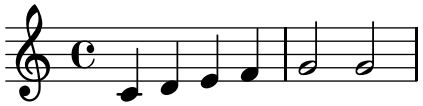
Inoltre, `@lilypondversion` mostra la versione di lilypond impiegata.

Quando si esegue `lilypond-book` su di esso, il risultato è un file Texinfo (con estensione '`.texi`') contenente degli elementi `@image` per l'HTML, Info e l'output per la stampa. `lilypond-book` genera le immagini della musica in formati EPS e PDF per l'utilizzo nell'output per la stampa e in formato PNG per l'utilizzo nell'output HTML e Info.

Vediamo due piccoli esempi. Un ambiente `lilypond`

```
@lilypond[fragment]
c' d' e' f' g'2 g'
@end lilypond
```

genera



La versione breve

```
@lilypond[fragment,staffsize=11]{<c' e' g'>}
```

genera



Diversamente da LaTeX, `@lilypond{...}` non genera un'immagine nel testo. Prende sempre un paragrafo proprio.

### 3.2.3 HTML

`lilypond-book` fornisce i seguenti comandi e ambienti per includere musica nei file HTML:

- il comando `\lilypond{...}`, dove si può inserire direttamente del codice lilypond corto
- l'ambiente `\begin{lilypond}...\end{lilypond}`, dove si può inserire direttamente del codice lilypond più lungo
- il comando `\lilypondfile{...}` per inserire un file lilypond
- il comando `\musicxmlfile{...}` per inserire un file MusicXML, che sarà elaborato da `musicxml2ly` e da `lilypond`.

Nel file di input, la musica viene specificata con uno dei seguenti comandi:

\begin{lilypond}[le,opzioni,vanno,qui] CODICE LILYPOND \end{lilypond}

\lilypond[le,opzioni,vanno,qui]{ CODICE LILYPOND }

\lilypondfile[le,opzioni,vanno,qui]{*nomefile*}

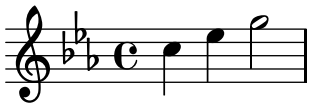
\musicxmlfile[le,opzioni,vanno,qui]{*nomefile*}

```
<lilypond le opzioni vanno qui>
  CODICE LILYPOND
</lilypond>

<lilypond le opzioni vanno qui: CODICE LILYPOND />

<lilypondfile le opzioni vanno qui>nomefile</lilypondfile>

<musicxmlfile le opzioni vanno qui>nomefile</musicxmlfile>
```

Ad esempio, puoi scrivere

```
<lilypond fragment relative=2>
\key c \minor c4 es g2
</lilypond>
```

`lilypond-book` genera quindi un file HTML con gli elementi appropriati per le immagini dei frammenti musicali:



Per le immagini in linea, si usa `<lilypond ... />`, dove le opzioni sono distinte dalla musica attraverso i due punti, ad esempio

```
Un po' di musica in <lilypond relative=2: a b c/> una linea di testo.
```

Per includere file separati, si usa

```
<lilypondfile opzione1 opzione2 ...>filename</lilypondfile>
```

`<musicxmlfile>` usa la stessa sintassi di `<lilypondfile>`, ma semplicemente si riferisce a un file MusicXML invece che a un file LilyPond.

Per una lista di opzioni da usare con gli elementi `lilypond` e `lilypondfile`, si veda ⟨undefined⟩ [Music fragment options], pagina ⟨undefined⟩.

Inoltre, `<lilypondversion/>` mostra la versione di lilypond impiegata.

### 3.2.4 DocBook

Per inserire frammenti di codice LilyPond è una buona idea mantenere la conformità del documento DocBook, in modo da poter usare gli editor DocBook, la validazione, etc. Quindi non si usano elementi personalizzati, ma si specifica una convenzione basata sugli elementi DocBook standard.

### Convenzioni comuni

Per inserire un frammento di qualsiasi tipo si usano gli elementi `mediaobject` e `inlinemediaobject`, in modo che il frammento possa essere formattato in linea o meno. Le opzioni di formattazione del frammento vengono sempre indicate nella proprietà `role` dell'elemento più interno (si vedano le prossime sezioni). Gli elementi sono scelti in modo da permettere agli editor DocBook di ottenere una formattazione ottimale. I file DocBook da elaborare con `lilypond-book` devono avere estensione '`.lyxml`'.

### Includere un file LilyPond

Si tratta del caso più semplice. Bisogna usare l'estensione '`.ly`' per il file da includere e inserirlo come uno standard `imageobject`, con la seguente struttura:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="music1.ly" role="printfilename" />
  </imageobject>
</mediaobject>
```

Nota che sei libero di usare `mediaobject` o `inlinemediaobject` come elemento più esterno.

### Includere codice LilyPond

È possibile includere codice LilyPond all'interno di un elemento `programlisting` in cui il linguaggio sia impostato su `lilypond` e con la seguente struttura:

```
<inlinemediaobject>
  <textobject>
    <programlisting language="lilypond" role="fragment verbatim staffsize=16 ragged-right re
\context Staff \with {
  \remove "Time_signature_engraver"
  \remove "Clef_engraver"}
  { c4( fis) }
    </programlisting>
  </textobject>
</inlinemediaobject>
```

Come si vede, l'elemento più esterno è `mediaobject` o `inlinemediaobject` e c'è un `textobject` che contiene al suo interno il `programlisting`.

### Elaborare il documento DocBook

L'esecuzione di `lilypond-book` su un file '`.lyxml`' creerà un documento DocBook valido con estensione '`.xml`' che potrà essere ulteriormente elaborato. Usando dblatex, creerà automaticamente un file PDF da questo documento. Per generare l'HTML (HTML Help, JavaHelp etc.) si possono usare i fogli di stile DocBook XSL ufficiali; tuttavia è possibile che sia necessario modificarli un po'.

## 3.3 Opzioni dei frammenti musicali

Nelle pagine che seguono, per 'comando LilyPond' si intende un qualsiasi comando descritto nelle sezioni precedenti che sia gestito da `lilypond-book` per produrre un frammento musicale. Per semplicità, i comandi LilyPond vengono mostrati soltanto nella sintassi LaTeX.

Nota che la stringa delle opzioni è analizzata da sinistra a destra; se un'opzione ricorre più di una volta, viene applicata nella sua ultima occorrenza.

Sono disponibili le seguenti opzioni per i comandi LilyPond:

`staffsize=`*altezza*
: Imposta la dimensione del pentagramma a *altezza*, misurata in punti.

`ragged-right`
: Produce linee con margine destro irregolare e spaziatura naturale, ovvero viene aggiunto `ragged-right = ##t` al frammento LilyPond. Frammenti con un solo rigo avranno sempre il margine destro irregolare, a meno che non venga specificato esplicitamente `noragged-right`.

`noragged-right`
: Per i frammenti di una sola linea, fa sì che la lunghezza del rigo venga estesa fino a coincidere con la larghezza della linea, ovvero viene aggiunto `ragged-right = ##f` al frammento LilyPond.

`line-width`
`line-width=`*dimensione*`\`*unità*
: Imposta la lunghezza della linea a *dimensione*, usando *unità* come unità di misura. *unità* può essere una delle seguenti stringhe: `cm`, `mm`, `in` o `pt`. Questa opzione riguarda l'output LilyPond (ovvero, la lunghezza del rigo del frammento musicale), non la formattazione del testo.

  Se usato senza un argomento, imposta la lunghezza della linea a un valore predefinito (calcolato da un algoritmo euristico).

  Se non viene assegnata un'opzione `line-width`, `lilypond-book` cerca di indovinare un valore predefinito per gli ambienti `lilypond` che non usano l'opzione `ragged-right`.

`papersize=`*stringa*
: Dove *stringa* è una delle dimensioni del foglio definite in '`scm/paper.scm`', ad esempio `a5`, `quarto`, `11x17` etc.

  I valori non definiti in '`scm/paper.scm`' saranno ignorati, sarà inviato un messaggio di avviso e al frammento sarà assegnata la dimensione predefinita, ovvero `a4`.

`notime` Non viene visualizzata l'indicazione di tempo e disabilita i segni relativi alla scansione ritmica (segno di tempo, barre di divisione) nella partitura.

`fragment` Fa sì che `lilypond-book` aggiunga del codice boilerplate in modo che sia possibile inserire semplicemente, ad esempio,

```
c'4
```

senza `\layout`, `\score`, etc.

`nofragment`
: Non aggiunge del codice ulteriore per completare il codice LilyPond nei frammenti musicali. Essendo l'impostazione predefinita, `nofragment` di norma è ridondante.

`indent=`*dimensione*`\`*unità*
: Imposta l'indentazione del primo sistema musicale a *dimensione*, usando *unità* come unità di misura. *unità* è una delle seguenti stringhe: `cm`, `mm`, `in` o `pt`. Questa opzione riguarda LilyPond, non la formattazione del testo.

`noindent` Imposta l'indentazione del primo sistema musicale su zero. Questa opzione interessa LilyPond, non la formattazione del testo. L'assenza di indentazione è l'impostazione predefinita, dunque normalmente `noindent` è ridondante.

`quote` Riduce la lunghezza della linea di un frammento musicale di $2 * 0.4\,\text{in}$ e inserisce l'output in un blocco per le citazioni. Il valore '0.4 in' può essere controllato con l'opzione `exampleindent`.

`exampleindent`
: Imposta la quantità di spazio con cui l'opzione `quote` indenta un frammento musicale.

`relative`
`relative=n`
: Usa la modalità di ottava relativa. Per impostazione predefinita, le altezze delle note sono riferite al Do centrale. L'argomento opzionale del numero intero specifica l'ottava della nota iniziale: il valore predefinito `1` è il Do centrale. L'opzione `relative` funziona solo quando è impostata l'opzione `fragment`, quindi `fragment` è implicitamente sottinteso da `relative`, a prescindere dalla presenza dell'opzione `(no)fragment` nel sorgente.

LilyPond usa `lilypond-book` anche per produrre la propria documentazione. A questo scopo, esistono altre opzioni più complesse per i frammenti musicali.

`verbatim` L'argomento di un comando LilyPond viene copiato nel file di output e racchiuso in un blocco di testo, seguito da qualsiasi testo assegnato con l'opzione `intertext` (non ancora implementato); quindi viene mostrata la musica vera e propria. Questa opzione non funziona bene con `\lilypond{}` se fa parte di un paragrafo.

Se `verbatim` viene usato in un comando `lilypondfile`, è possibile includere il testo di una parte soltanto del file sorgente. Se il file sorgente ha un commento contenente '`begin verbatim`' (senza virgolette), la citazione del sorgente nel blocco testuale inizierà dopo l'ultima occorrenza di tale commento; in modo analogo, la citazione del testo sorgente si fermerà proprio prima della prima occorrenza di un commento contenente '`end verbatim`', se presente. Nel seguente file sorgente di esempio, la musica viene interpretata in modalità relativa ma il blocco testuale non mostrerà il blocco `relative`, ovvero

```
\relative c' { % begin verbatim
  c4 e2 g4
  f2 e % end verbatim
}
```

mostrerà il seguente blocco testuale

```
  c4 e2 g4
  f2 e
```

Se si desidera tradurre i commenti e i nomi delle variabili nell'output verbatim ma non nei sorgenti, si può impostare la variabile d'ambiente `LYDOC_LOCALEDIR` sul percorso di una directory; la directory deve contenere un albero dei cataloghi di messaggio '`.mo`' che hanno `lilypond-doc` come dominio.

`addversion`
: (Solo per l'output Texinfo.) Aggiunge `\version @w{"@version{}"}` nella prima riga dell'output di `verbatim`.

`texidoc` (Solo per l'output Texinfo.) Se `lilypond` viene lanciato con l'opzione '`--header=texidoc`' e il file da elaborare si chiama '`foo.ly`', verrà creato un file '`foo.texidoc`' a patto che ci sia un campo `texidoc` nel blocco `\header`. L'opzione `texidoc` fa sì che `lilypond-book` includa tali file, aggiungendo il loro contenuto in forma di blocco di documentazione proprio prima del frammento di musica (ma fuori dall'ambiente `example` generato da un'opzione `quote`).

Se il file '`foo.ly`' contiene

```
\header {
  texidoc = "Questo file mostra il funzionamento di una singola nota."
```

```
}
{ c'4 }
```

e il documento Texinfo ‘test.texinfo’ contiene

```
@lilypondfile[texidoc]{foo.ly}
```

il seguente comando produce il risultato atteso

```
lilypond-book --pdf --process="lilypond \
  -dbackend=eps --header=texidoc" test.texinfo
```

Per la maggior parte, i documenti di test di LilyPond (nella directory ‘input’ della distribuzione) sono piccoli file ‘.ly’ che hanno esattamente questo aspetto.

Ai fini della localizzazione, se il documento Texinfo document contiene `@documentlanguage` *LANG* e l'header di ‘foo.ly’ contiene un campo `texidoc`*LANG*, quando si lancia `lilypond` con l'opzione ‘`--header=texidoc`*LANG*’ verrà incluso ‘`foo.texidoc`*LANG*’ invece di ‘foo.texidoc’.

`doctitle` (Solo per l'output Texinfo.) Questa opzione funziona in modo simile all'opzione `texidoc`: se `lilypond` viene lanciato con l'opzione ‘`--header=doctitle`’ e il file da elaborare si chiama ‘foo.ly’ e contiene un campo `doctitle` nel blocco `\header`, viene creato un file ‘foo.doctitle’. Se si usa l'opzione `doctitle`, i contenuti di ‘foo.doctitle’, che dovrebbero trovarsi su una singola linea di *text*, vengono inseriti nel documento Texinfo come `@lydoctitle` *text*. `@lydoctitle` è una macro definita nel documento Texinfo. Lo stesso discorso relativo all'elaborazione `texidoc` delle lingue localizzate si applica anche a `doctitle`.

`nogettext`
(Solo per l'output Texinfo.) Non tradurre i commenti e i nomi delle variabili nel blocco testuale del frammento citato.

`printfilename`
Se un file di input di LilyPond viene incluso con `\lilypondfile`, il nome del file viene mostrato immediatamente prima del frammento musicale. Per l'output HTML, questo nome è un collegamento. Viene mostrata solo la base del nome del file, ovvero viene tolta la parte che costituisce il percorso del file.

## 3.4 Utilizzo di `lilypond-book`

`lilypond-book` crea un file con una delle seguenti estensioni: ‘.tex’, ‘.texi’, ‘.html’ o ‘.xml’, a seconda del formato dell'output. Tutti i file ‘.tex’, ‘.texi’ e ‘.xml’ necessitano di un'ulteriore elaborazione.

### Istruzioni specifiche di ogni formato

#### LaTeX

Esistono due modi di elaborare il documento LaTeX per la stampa o la pubblicazione: generare direttamente un file PDF tramite PDFLaTeX oppure generare un file PostScript tramite LaTeX, attraverso un traduttore da DVI a PostScript come `dvips`. Il primo modo è più semplice e raccomandato[1], e indipendentemente da quello che userai, puoi convertire facilmente PostScript e PDF con strumenti come `ps2pdf` e `pdf2ps` inclusi nel pacchetto Ghostscript.

Per creare un file PDF con PDFLaTeX, si usa

```
lilypond-book --pdf tuofile.lytex
pdflatex tuofile.tex
```

[1] Nota che PDFLaTeX e LaTeX potrebbero non essere entrambi utilizzabili per compilare un qualsiasi documento LaTeX: ecco perché vengono illustrati i due modi.

Per produrre l'output PDF attraverso LaTeX/dvips/ps2pdf, bisogna usare questi comandi

```
lilypond-book tuofile.lytex
latex tuofile.tex
dvips -Ppdf tuofile.dvi
ps2pdf tuofile.ps
```

Il file '.dvi' creato da questa sequenza non conterrà le teste delle note. È normale; se si seguono le istruzioni, le teste verranno incluse nei file '.ps' e '.pdf'.

L'esecuzione di dvips potrebbe generare dei messaggi di avviso relativi ai caratteri; questi messaggi sono innocui e possono essere ignorati. Se esegui latex in modalità due colonne, ricorda di aggiungere '-t landscape' alle opzioni di dvips.

## Problemi noti e avvertimenti

Il comando \pageBreak non funziona all'interno dell'ambiente \begin{lilypond} ... \end{lilypond}.

Molte variabili del blocco \paper non funzionano all'interno dell'ambiente \begin{lilypond} ... \end{lilypond}. Usa \newcommand con \betweenLilyPondSystem nel preambolo;

```
\newcommand{\betweenLilyPondSystem}[1]{\vspace{36mm}\linebreak}
```

## Texinfo

Per generare un documento Texinfo (in qualsiasi formato di output), si seguono le normali procedure usate con Texinfo; ovvero, si lancia texi2pdf o texi2dvi o makeinfo, a seconda del formato di output che si vuole creare. Si veda la documentazione di Texinfo per ulteriori dettagli.

## Opzioni da linea di comando

lilypond-book accetta le seguenti opzioni da linea di comando:

`-f` *formato*
`--format=`*formato*
: Specifica il tipo di documento da elaborare: html, latex, texi (il formato predefinito) o docbook. Se manca questa opzione, lilypond-book cerca di rilevare il formato automaticamente, si veda ⟨undefined⟩ [Filename extensions], pagina ⟨undefined⟩. Attualmente, texi è equivalente a texi-html.

`-F` *filtro*
`--filter=`*filtro*
: Convoglia i frammenti attraverso il *filtro*. lilypond-book non esegue contemporaneamente il filtro e l'elaborazione. Ad esempio,

  ```
  lilypond-book --filter='convert-ly --from=2.0.0 -' mio-libro.tely
  ```

`-h`
`--help`
: Mostra un breve messaggio di aiuto.

`-I` *dir*
`--include=`*dir*
: Aggiunge *dir* al percorso di inclusione. lilypond-book cerca anche dei frammenti già compilati nel percorso di inclusione e non li riscrive nella directory di output, quindi in alcuni casi è necessario eseguire ulteriori comandi come makeinfo o latex con le stesse opzioni '-I *dir*'.

`-l` *loglevel*
`--loglevel=`*loglevel*
: Imposta la verbosità dell'output su *loglevel*. I valori possibili sono NONE, ERROR, WARNING, PROGRESS (predefinito) e DEBUG. Se questa opzione non viene usata e la

variabile d'ambiente `LILYPOND_BOOK_LOGLEVEL` è impostata, il suo valore viene usato come loglevel.

`-o` *dir*
`--output=`*dir*
: Salva i file generati nella directory *dir*. L'esecuzione di `lilypond-book` genera tanti piccoli file che LilyPond elaborerà. Per evitare tutto questo disordine nella directory dei sorgenti, si usa l'opzione da linea di comando '`--output`' e si entra in questa directory prima di eseguire `latex` o `makeinfo`.

```
lilypond-book --output=out tuofile.lytex
cd out
...
```

`--skip-lily-check`
: Non si arresta se non viene trovato l'output di lilypond. Viene usata per la documentazione Info di LilyPond, che è priva di immagini.

`--skip-png-check`
: Non si arresta se non vengono trovate immagini PNG per i file EPS. Viene usata per la documentazione Info di LilyPond, che è priva di immagini.

`--lily-output-dir=`*dir*
: Scrive i file lily-XXX nella directory *dir*, crea un link nella directory '`--output`'. Si usa questa opzione per risparmiare tempo nella compilazione di documenti situati in directory diverse che condividono molti identici frammenti.

`--lily-loglevel=`*loglevel*
: Set the output verbosity of the invoked `lilypond` calls to *loglevel*. I valori possibili sono `NONE`, `ERROR`, `WARNING`, `BASIC_PROGRESS`, `PROGRESS`, `INFO` (predefinito) e `DEBUG`. Se questa opzione non viene usata e la variabile d'ambiente `LILYPOND_LOGLEVEL` è impostata, il suo valore viene usato come loglevel.

`--info-images-dir=`*dir*
: Formatta l'output di Texinfo in modo che Info cerchi le immagini della musica in *dir*.

`--latex-program=`*prog*
: Lancia l'eseguibile `prog` invece di `latex`. Questa opzione è utile, ad esempio, se il documento è elaborato con `xelatex`.

`--left-padding=`*quantità*
: Crea una spaziatura corrispondente a questa quantità tra i riquadri EPS. *quantità* è misurata in millimetri e il valore predefinito è 3.0. Questa opzione si usa se i righi dello spartito oltrepassano il margine destro.

  La larghezza di un sistema molto denso può variare in base agli elementi della notazione attaccati al margine sinistro, come i numeri di battuta e i nomi degli strumenti. Questa opzione accorcia tutte le linee e le sposta a a destra della stessa quantità.

`-P` *comando*
`--process=`*comando*
: Elabora i frammenti di LilyPond con *comando*. Il comando predefinito è `lilypond`. `lilypond-book` non userà '`--filter`' e '`--process`' contemporaneamente.

`--pdf` Crea file PDF da usare con PDFLaTeX.

`--redirect-lilypond-output`
: Per impostazione predefinita, l'output viene mostrato sul terminale. Questa opzione redirige tutto l'output in dei file di log nella stessa directory dei file sorgente.

`--use-source-file-names`
: Salva i file di output dei frammenti con lo stesso nome, esclusa l'estensione, dei sorgenti. Questa opzione funziona solo con i frammenti inclusi con `lilypondfile` e solo se le directory indicate da '`--output-dir`' e '`--lily-output-dir`' sono diverse.

`-V`
`--verbose`
: Mostra un output dettagliato. Questo è equivalente a `--loglevel=DEBUG`.

`-v`
`--version`
: Mostra informazioni sulla versione.

### Problemi noti e avvertimenti

Il comando Texinfo `@pagesizes` non viene inrerpretato. Allo stesso modo, i comandi LaTeX che modificano i margini e la larghezza della linea dopo il preambolo vengono ignorati.

Solo il primo `\score` di un blocco LilyPond viene elaborato.

## 3.5 Estensioni dei nomi di file

Si può usare qualsiasi estensione per il file di input, ma se non si usa l'estensione raccomandata per uno specifico formato potrebbe essere necessario specificare a mano il formato di output; per i dettagli si veda ⟨undefined⟩ [Invoking lilypond-book], pagina ⟨undefined⟩. Altrimenti, `lilypond-book` sceglie automaticamente il formato di output in base all'estensione del file di input.

| **estensione** | **formato di output** |
|---|---|
| '`.html`' | HTML |
| '`.htmly`' | HTML |
| '`.itely`' | Texinfo |
| '`.latex`' | LaTeX |
| '`.lytex`' | LaTeX |
| '`.lyxml`' | DocBook |
| '`.tely`' | Texinfo |
| '`.tex`' | LaTeX |
| '`.texi`' | Texinfo |
| '`.texinfo`' | Texinfo |
| '`.xml`' | HTML |

Se si usa per il file di input la stessa estensione che `lilypond-book` usa per il file di output e se il file di input è nella stessa directory in cui lavora `lilypond-book`, bisogna usare l'opzione '`--output`' per far sì che `lilypond-book` sia eseguito; altrimenti si ferma e mostra un messaggio di errore simile a "L'output sovrascriverebbe il file di input".

## 3.6 Modelli per lilypond-book

Ecco alcuni modelli da usare con `lilypond-book`. Se non hai familiarità con questo programma, consulta Capitolo 3 [lilypond-book], pagina 19.

### 3.6.1 LaTeX

Si possono includere frammenti LilyPond in un documento LaTeX.

```
\documentclass[]{article}

\begin{document}

Normale testo LaTeX.

\begin{lilypond}
\relative c'' {
  a4 b c d
}
\end{lilypond}

Altro testo LaTeX, seguito da alcune opzioni tra parentesi quadre.

\begin{lilypond}[fragment,relative=2,quote,staffsize=26,verbatim]
d4 c b a
\end{lilypond}
\end{document}
```

### 3.6.2 Texinfo

Si possono includere frammenti LilyPond in Texinfo; infatti questo intero manuale è scritto in Texinfo.

```
\input texinfo @node Top
@top

Testo Texinfo

@lilypond
\relative c' {
  a4 b c d
}
@end lilypond

Altro testo Texinfo, seguito dalle opzioni tra parentesi.

@lilypond[verbatim,fragment,ragged-right]
d4 c b a
@end lilypond

@bye
```

### 3.6.3 html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- header_tag -->
<HTML>
<body>

<p>
I documenti per lilypond-book possono combinare liberamente musica e testo.  Ad
esempio,
<lilypond>
```

```
\relative c'' {
  a4 b c d
}
</lilypond>
</p>

<p>
Ancora un po' di Lilypond, questa volta con delle opzioni:

<lilypond fragment quote staffsize=26 verbatim>
a4 b c d
</lilypond>
</p>

</body>
</html>
```

### 3.6.4 xelatex

```
\documentclass{article}
\usepackage{ifxetex}
\ifxetex
%elementi specifici di xetex
\usepackage{xunicode,fontspec,xltxtra}
\setmainfont[Numbers=OldStyle]{Times New Roman}
\setsansfont{Arial}
\else
%Questo può essere lasciato vuoto se non si usa pdftex
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{mathptmx}%Times
\usepackage{helvet}%Helvetica
\fi
%Qui è possibile inserire tutti i pacchetti inclusi anche in pdftex
\usepackage[ngerman,finnish,english]{babel}
\usepackage{graphicx}

\begin{document}
\title{Un breve documento con LilyPond e xelatex}
\maketitle

I comandi abituali di \textbf{font} interni al \emph{testo} funzionano,
perché \textsf{sono supportati da \LaTeX{} e XeteX.}
Se vuoi usare comandi specifici come \verb+\XeTeX+, devi
includerli nuovamente in un ambiente \verb+\ifxetex+.
You can use this to print the \ifxetex \XeTeX{} command \else
XeTeX command \fi which is not known to normal \LaTeX .

Nel testo normale si possono usare semplicemente i comandi LilyPond, come in
questo esempio:
```

```
\begin{lilypond}
{a2 b c'8 c' c' c'}
\end{lilypond}

\noindent
e così via.

I tipi di carattere dei frammenti inseriti con LilyPond devono essere impostati
all'interno dei frammenti stessi.  Si legga il manuale di Uso dell'applicazione per
sapere come usare lilypond-book.

\selectlanguage{ngerman}
Auch Umlaute funktionieren ohne die \LaTeX -Befehle, wie auch alle
anderen
seltsamen Zeichen: __ ______, wenn sie von der Schriftart
unterst__tzt werden.
\end{document}
```

## 3.7 Condividere l'indice

Queste funzioni sono già incluse nel pacchetto OrchestralLily:

http://repo.or.cz/w/orchestrallily.git

Alcuni utenti preferiscono esportare l'indice da lilypond e leggerlo da dentro LaTeX per la sua maggiore flessibilità nella gestione del testo.

### Esportare l'indice da LilyPond

Per questo esempio si presume che lo spartito abbia vari movimenti nello stesso file di output di lilypond.

```
#(define (oly:create-toc-file layout pages)
  (let* ((label-table (ly:output-def-lookup layout 'label-page-table)))
    (if (not (null? label-table))
      (let* ((format-line (lambda (toc-item)
             (let* ((label (car toc-item))
                    (text  (caddr toc-item))
                    (label-page (and (list? label-table)
                                     (assoc label label-table)))
                    (page (and label-page (cdr label-page))))
               (format #f "~a, section, 1, {~a}, ~a" page text label))))
             (formatted-toc-items (map format-line (toc-items)))
             (whole-string (string-join formatted-toc-items ",\n"))
             (output-name (ly:parser-output-name parser))
             (outfilename (format "~a.toc" output-name))
             (outfile (open-output-file outfilename)))
        (if (output-port? outfile)
            (display whole-string outfile)
            (ly:warning (_ "Unable to open output file ~a for the TOC information") outfilename))
        (close-output-port outfile)))))

\paper {
  #(define (page-post-process layout pages) (oly:create-toc-file layout pages))
}
```

### Importare l'indice in LaTeX

In LaTeX l'intestazione deve includere:

```
\usepackage{pdfpages}
```

```
\includescore{nameofthescore}
```

dove `\includescore` viene definito in questo modo:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \includescore{PossibleExtension}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read in the TOC entries for a PDF file from the corresponding .toc file.
% This requires some heave latex tweaking, since reading in things from a file
% and inserting it into the arguments of a macro is not (easily) possible

% Solution by Patrick Fimml on #latex on April 18, 2009:
% \readfile{filename}{\variable}
% reads in the contents of the file into \variable (undefined if file
% doesn't exist)
\newread\readfile@f
\def\readfile@line#1{%
{\catcode`\^^M=10\global\read\readfile@f to \readfile@tmp}%
\edef\do{\noexpand\g@addto@macro{\noexpand#1}{\readfile@tmp}}\do%
\ifeof\readfile@f\else%
\readfile@line{#1}%
\fi%
}
\def\readfile#1#2{%
\openin\readfile@f=#1 %
\ifeof\readfile@f%
\typeout{No TOC file #1 available!}%
\else%
\gdef#2{}%
\readfile@line{#2}%
\fi
\closein\readfile@f%
}%


\newcommand{\includescore}[1]{
\def\oly@fname{\oly@basename\@ifmtarg{#1}{}{_#1}}
\let\oly@addtotoc\undefined
\readfile{\oly@xxxxxxxxx}{\oly@addtotoc}
\ifx\oly@addtotoc\undefined
\includepdf[pages=-]{\oly@fname}
\else
\edef\includeit{\noexpand\includepdf[pages=-,addtotoc={\oly@addtotoc}]
{\oly@fname}}\includeit
\fi
}
```

## 3.8 Metodi alternativi per combinare testo e musica

Altri modi per combinare testo e musica (senza usare `lilypond-book`) sono trattati in ⟨undefined⟩ [LilyPond output in other programs], pagina ⟨undefined⟩.

# 4 Programmi esterni

LilyPond può interagire con altri programmi in vari modi.

## 4.1 Punta e clicca

Il "punta e clicca" aggiunge dei collegamenti ai documenti pdf per certi elementi musicali.

### 4.1.1 Configurare il sistema

Quando questa funzionalità è attiva, LilyPond aggiunge dei collegamenti ipertestuali al file PDF. Questi collegamenti vengono inviati a un 'programma di supporto per URI' o al browser web, che apre un editor di testo col cursore posizionato nel punto giusto.

Perché questo procedimento funzioni è necessario configurare il lettore PDF in modo che segua i collegamenti ipertestuali usando lo script '`lilypond-invoke-editor`' fornito insieme a LilyPond.

'`lilypond-invoke-editor`' è un piccolo programma di supporto. Lancia un editor per gli URI `textedit` e un browser web per altri URI. Valuta la variabile d'ambiente `EDITOR` in base a questi schemi,

`emacs` verrà quindi lanciato il comando

`emacsclient --no-wait +`*`line`*`:`*`column file`*

`gvim` verrà quindi lanciato il comando

`gvim --remote +:`*`line`*`:norm`*`column file`*

`nedit` verrà quindi lanciato il comando

`nc -noask +`*`line file`*`'`

La variabile d'ambiente `LYEDITOR` ha la precedenza sulla variabile `EDITOR`. Contiene il comando per lanciare l'editor, dove `%(file)s`, `%(column)s`, `%(line)s` vengono sostituiti rispettivamente dal file, dalla colonna e dalla riga. L'impostazione

```
emacsclient --no-wait +%(line)s:%(column)s %(file)s
```

per `LYEDITOR` è equivalente alla chiamata standard emacsclient.

### Usare Xpdf

Se si usa Xpdf su UNIX, si deve inserire la seguente riga nel file '`xpdfrc`'. Su UNIX, questo file può essere '`/etc/xpdfrc`' oppure '`$HOME/.xpdfrc`'.

```
urlCommand      "lilypond-invoke-editor %s"
```

Se si usa Ubuntu, è probabile che la versione di Xpdf installata nel sistema causi il crash per qualsiasi file PDF: questa situazione continua da molti anni ed è dovuta a una corrispondenza sbagliata tra librerie. Conviene installare un pacchetto aggiornato di '`xpdf`' e il corrispondente pacchetto '`libpoppler`' da Debian. Dopo aver verificato che funziona, si può usare il comando

```
sudo apt-mark hold xpdf
```

per impedire a Ubuntu di sovrascriverlo al prossimo 'aggiornamento' del suo pacchetto difettoso.

### Usare GNOME 2

Per usare GNOME 2 (e i visualizzatori PDF ad esso integrati), il magico comando che fornisce al sistema gli URI '`textedit:`' è

```
gconftool-2 -t string -s /desktop/gnome/url-handlers/textedit/command "lilypond-invoke-edito
gconftool-2 -s /desktop/gnome/url-handlers/textedit/needs_terminal false -t bool
gconftool-2 -t bool -s /desktop/gnome/url-handlers/textedit/enabled true
```

Dopo questi comandi,

```
gnome-open textedit:///etc/issue:1:0:0
```

dovrebbe lanciare '`lilypond-invoke-editor`' per l'apertura del file.

### Usare GNOME 3

In GNOME 3, gli URI sono gestiti da 'gvfs' invece che da 'gconf'. Si crea un file in una directory locale (ad esempio '`/tmp`') che abbia il nome '`lilypond-invoke-editor.desktop`' e il seguente contenuto

```
[Desktop Entry]
Version=1.0
Name=lilypond-invoke-editor
GenericName=Textedit URI handler
Comment=URI handler for textedit:
Exec=lilypond-invoke-editor %u
Terminal=false
Type=Application
MimeType=x-scheme-handler/textedit;
Categories=Editor
NoDisplay=true
```

e poi si eseguono i comandi

```
xdg-desktop-menu install ./lilypond-invoke-editor.desktop
xdg-mime default lilypond-invoke-editor.desktop x-scheme-handler/textedit
```

Dopo questi comandi,

```
gnome-open textedit:///etc/issue:1:0:0
```

dovrebbe lanciare '`lilypond-invoke-editor`' per l'apertura del file.

### Ulteriore configurazione per Evince

Se `gnome-open` funziona, ma Evince si rifiuta ancora di aprire i collegamenti punta e clicca a causa di permessi negati, potrebbe essere necessario cambiare il profilo Apparmor di Evince che controlla il tipo di azioni che Evince ha il permesso di eseguire.

In Ubuntu, si modifica il file '`/etc/apparmor.d/local/usr.bin.evince`' e si aggiungono le seguenti righe:

```
# Per i collegamenti Textedit
/usr/local/bin/lilypond-invoke-editor Cx -> sanitized_helper,
```

Dopo aver aggiunto queste righe, si lancia il comando

```
sudo apparmor_parser -r -T -W /etc/apparmor.d/usr.bin.evince
```

Ora Evince dovrebbe essere in grado di aprire i collegamenti punta e clicca. È probabile che configurazioni simili funzionino anche con altri visualizzatori.

### Abilitare il punta e clicca

La funzionalità "punta e clicca" è abilitata di default quando si creano i file PDF.

I collegamenti "punta e clicca" appesantiscono sensibilmente i file di output. Per ridurre la dimensione dei file PDF e PS, è possibile disattivare il "punta e clicca" inserendo

```
\pointAndClickOff
```

in un file '`.ly`'. Il "punta e clicca" può essere abilitato esplicitamente con

```
\pointAndClickOn
```

Si può disabilitare il "punta e clicca" anche con un'opzione da linea di comando:

```
lilypond -dno-point-and-click file.ly
```

**Nota:** Occorre sempre disattivare il "punta e clicca" nei file LilyPond che si vogliano diffondere, per evitare di includere nel file .pdf delle informazioni sui percorsi del proprio computer: questo infatti può costituire un rischio di sicurezza.

### Punta e clicca selettivo

Per alcune applicazioni interattive, si potrebbe voler includere soltanto alcuni elementi punta e clicca. Ad esempio, se qualcuno volesse creare un'applicazione che riproduca audio o video a partire da una nota in particolare, sarebbe inopportuno che il clic sulla nota portasse alla posizione di un'alterazione o di una legatura che si trovi sopra quella nota.

Questo può essere controllato indicando quali eventi includere:

- Codice interno al file '`.ly`':

```
\pointAndClickTypes #'note-event
\relative c' {
  c2\f( f)
}
```

oppure

```
#(ly:set-option 'point-and-click 'note-event)
\relative c' {
  c2\f( f)
}
```

- Linea di comando:

```
lilypond -dpoint-and-click=note-event   example.ly
```

Si può includere più di un evento:

- Codice interno al file '`.ly`':

```
\pointAndClickTypes #'(note-event dynamic-event)
\relative c' {
  c2\f( f)
}
```

oppure

```
#(ly:set-option 'point-and-click '(note-event dynamic-event))
\relative c' {
  c2\f( f)
}
```

- Linea di comando:

```
lilypond \
  -e"(ly:set-option 'point-and-click '(note-event dynamic-event))" \
  example.ly
```

## 4.2 LilyPond e gli editor di testo

Vari editor di testo hanno funzionalità specifiche per LilyPond.

### Modalità di Emacs

Emacs ha una modalità '`lilypond-mode`', che fornisce il completamento delle parole, l'indentazione, le parentesi automatiche e la colorazione della sintassi specifiche di LilyPond, comode scorciatoie per la compilazione e la possibilità di leggere i manuali di LilyPond usando Info. Se '`lilypond-mode`' non è installato nel tuo computer, vedi sotto.

Una modalità Emacs per inserire la musica e eseguire LilyPond è presente nell'archivio dei sorgenti nella directory '`elisp`'. Lancia `make install` per installarla in *elispdir*. Il file '`lilypond-init.el`' deve essere messo in *load-path*'`/site-start.d/`' o aggiunto a '`~/.emacs`' oppure '`~/.emacs.el`'.

Come utente normale, puoi aggiungere il percorso dei sorgenti (ad esempio '`~/site-lisp/`') al tuo *load-path* aggiungendo la seguente riga (modificata di conseguenza) al file '`~/.emacs`'

```
(setq load-path (append (list (expand-file-name "~/site-lisp")) load-path))
```

### Modalità di Vim

Per Vim, sono disponibili le seguenti funzionalità per LilyPond: un plugin di riconoscimento del tipo di file, una modalità di indentazione e di evidenziazione della sintassi. Per abilitarle, crea (o modifica) il file '`$HOME/.vimrc`' in modo che contenga queste tre righe, in questo ordine:

```
filetype off
set runtimepath+=/usr/local/share/lilypond/current/vim/
filetype on
```

Se LilyPond non è installato nella directory '`/usr/local/`', modifica il percorso in modo adeguato. Questo argomento è trattato in Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

### Altri editor

Altri editor (sia testuali che grafici) supportano LilyPond, ma i loro specifici file di configurazione non sono distribuiti insieme a LilyPond. Consulta la documentazione di questi programmi per maggiori informazioni. Questi editor sono elencati in Sezione "Editing facilitato" in *Informazioni generali*.

## 4.3 Conversione da altri formati

È possibile inserire la musica anche importandola da altri formati. Questo capitolo documenta gli strumenti inclusi nella distribuzione che svolgono questo compito. Esistono altri strumenti che producono l'input di LilyPond, ad esempio i sequencer ad interfaccia grafica e i convertitori XML. Per maggiori dettagli consulta il sito web.

Si tratta di programmi separati da `lilypond` e sono eseguiti dalla linea di comando; si veda ⟨undefined⟩ [Command-line usage], pagina ⟨undefined⟩ per maggiori informazioni. Se usi MacOS 10.3 o 10.4 e hai problemi a eseguire alcuni di questi script, ad esempio `convert-ly`, vedi Sezione "MacOS X" in *Informazioni generali*.

### Problemi noti e avvertimenti

Purtroppo non abbiamo le risorse per mantenere questi programmi; prendeteli "così come sono"! Accettiamo con piacere le *patch*, ma ci sono poche possibilità che i bug vengano risolti.

### 4.3.1 Utilizzo di `midi2ly`

`midi2ly` trasforma un file MIDI Type 1 in un file sorgente di LilyPond.

Il protocollo MIDI (Music Instrument Digital Interface) è uno standard per gli strumenti digitali: fornisce le specifiche per la connessione via cavo, un protocollo seriale e un formato di file. Il formato MIDI è uno standard de facto per esportare la musica da altri programmi, dunque questa capacità diventa utile quando si importano file creati con un programma che converta direttamente in questo formato.

`midi2ly` converte le tracce presenti nei contesti Sezione "Staff" in *Guida al Funzionamento Interno* e i canali dei contesti Sezione "Voice" in *Guida al Funzionamento Interno*. Per indicare le altezze viene usata la modalità relativa, mentre le durate sono precisate solo quando necessario.

È possibile registrare un file MIDI usando una tastiera digitale e poi convertirlo in file '`.ly`'. Tuttavia, la conversione da MIDI a LY non è banale: l'esecuzione umana non sarà mai sufficientemente precisa dal punto di vista ritmico. Se lanciata con la quantizzazione (opzioni '`-s`' e '`-d`') `midi2ly` cerca di compensare questi errori di tempo, ma non è molto efficace. Dunque non si raccomanda l'uso di `midi2ly` per i file midi generati a partire da un'esecuzione umana.

Si lancia dalla linea di comando in questo modo:

`midi2ly [`*opzione*`]... `*file-midi*

Attenzione: per 'linea di comando' si intende la linea di comando del sistema operativo. Si veda ⟨undefined⟩ [Converting from other formats], pagina ⟨undefined⟩ per maggiori informazioni su questo argomento.

`midi2ly` accetta le seguenti opzioni.

`-a, --absolute-pitches`
: Crea altezze assolute.

`-d, --duration-quant=`*DUR*
: Quantizza la durata delle note di *DUR*.

`-e, --explicit-durations`
: Crea durate esplicite.

`-h,--help`
: Mostra una sintesi dell'utilizzo del programma.

`-k, --key=`*acc*`[:`*minor*`]`
: Imposta la tonalità predefinita. $acc > 0$ imposta il numero di diesis; $acc < 0$ imposta il numero di bemolle. Una tonalità minore si indica con `:1`.

`-o, --output=`*file*
: Scrive l'output in *file*.

`-s, --start-quant=`*DUR*
: Quantize note starts on *DUR*.

`-t, --allow-tuplet=`*DUR*NUM/DEN*
: Consente l'inserimento di gruppi irregolari *DUR*NUM/DEN*.

`-v, --verbose`
: Mostra un output dettagliato.

`-V, --version`
: Mostra il numero di versione.

`-w, --warranty`
: Mostra la garanzia e il copyright.

`-x, --text-lyrics`
: Interpreta il testo come liriche.

## Problemi noti e avvertimenti

Le note sovrapposte in un arpeggio non sono rese correttamente: viene letta solo la prima nota, mentre le altre vengono ignorate. Assegna a tutte la stessa durata e introduci le opportune indicazioni di fraseggio o di pedalizzazione.

### 4.3.2 Utilizzo di `musicxml2ly`

MusicXML è un dialetto di XML che viene usato per rappresentare la notazione musicale.

`musicxml2ly` estrae le note, le articolazioni, la struttura della partitura, il testi, etc. da file MusicXML organizzati in parti; quindi li scrive in un file '`.ly`'. Si usa dalla linea di comando.

Si lancia dalla linea di comando nel modo seguente,

`musicxml2ly [`*`opzione`*`]... `*`file-xml`*

Attenzione: per 'linea di comando' si intende la linea di comando del sistema operativo. Si veda ⟨undefined⟩ [Converting from other formats], pagina ⟨undefined⟩ per maggiori informazioni su questo argomento.

Se il nome del file è '`-`', `musicxml2ly` legge l'input dalla linea di comando.

`musicxml2ly` accetta le seguenti opzioni:

`-a, --absolute`
: converte le altezze relative in assolute.

`-h,--help`
: mostra una sintesi dell'utilizzo e delle opzioni.

`-l, --language=LANG`
: usa LANG per i nomi delle altezze, ad esempio 'deutsch' per i nomi delle note in tedesco.

`--loglevel=`*`loglevel`*
: Imposta la verbosità dell'output su *loglevel*. I valori possibili sono `NONE`, `ERROR`, `WARNING`, `PROGRESS` (predefinito) e `DEBUG`.

`--lxml`
: usa il pacchetto Python lxml.etree per l'analisi della sintassi XML; usa meno memoria e tempo del processore.

`-m, --midi`
: attiva il blocco midi.

`--nd --no-articulation-directions`
: non converte le direzioni (`^`, `_` o `-`) per articolazioni, dinamiche, etc.

`--no-beaming`
: ignora le informazioni relative alle travature, impiegando la disposizione automatica delle travature fornita da LilyPond.

`-o,--output=`*`file`*
: imposta il nome del file di output su *file*. Se *file* è '`-`', l'output sarà salvato su stdout. Se non specificato, verrà usato *file-xml*'`.ly`'.

`-r,--relative`
: converte le altezze in modalità relativa (predefinito).

`-v, --verbose`
: Mostra un output dettagliato.

`--version`
: Mostra informazioni sulla versione.

`-z,--compressed`
: il file di input è un file MusicXML compresso in un archivio ZIP.

### 4.3.3 Utilizzo di `abc2ly`

**Nota:** Questo programma non è supportato e potrebbe essere rimosso dalle future versioni di LilyPond.

ABC è un semplice formato basato su ASCII. È descritto nel sito di ABC:
http://www.walshaw.plus.com/abc/learn.html.

`abc2ly` traduce dal formato ABC al formato LilyPond. Viene lanciato nel modo seguente:

`abc2ly [`*opzione*`]... `*file-abc*

`abc2ly` accetta le seguenti opzioni:

`-b,--beams=None`
: preserva le regole di disposizione delle travature di ABC

`-h,--help`
: mostra questo messaggio di aiuto

`-o,--output=`*file*
: imposta il nome del file di output su *file*.

`-s,--strict`
: imposta una modalità di interpretazione letterale per effettuare una conversione stretta

`--version`
: mostra informazioni sulla versione.

Esiste una rudimentale funzione per aggiungere codice LilyPond nel file sorgente ABC. Se scrivi:

```
%%LY voices \set autoBeaming = ##f
```

il testo che segue la parola chiave 'voices' verrà inserito nella voce in uso del file di output LilyPond.

Analogalmente,

```
%%LY slyrics more words
```

fa sì che il testo che segue la parola chiave 'slyrics' venga inserito nella riga corrente del testo.

#### Problemi noti e avvertimenti

Lo standard ABC standard non è molto 'standard'. Per le funzionalità più avanzate (ad esempio, la musica polifonica) esistono diversi tipi di convenzioni.

Un file che contiene più di un brano non può essere convertito.

ABC allinea le parole e le note all'inizio di una riga; `abc2ly` non lo fa.

`abc2ly` ignora la disposizione delle travature fatta da ABC.

### 4.3.4 Utilizzo di `etf2ly`

**Nota:** Questo programma non è supportato e potrebbe essere rimosso dalle future versioni di LilyPond.

ETF (Enigma Transport Format) è un formato usato da Finale, un prodotto di Coda Music Technology. `etf2ly` converte parte di un file ETF in un file LilyPond pronto all'uso.

Si lancia dalla linea di comando nel modo seguente.

`etf2ly [`*`opzione`*`]... `*`file-etf`*

Attenzione: per 'linea di comando' si intende la linea di comando del sistema operativo. Si veda ⟨undefined⟩ [Converting from other formats], pagina ⟨undefined⟩ per maggiori informazioni su questo argomento.

`etf2ly` accetta le seguenti opzioni:

`-h,--help`
: mostra questo messaggio di aiuto

`-o,--output=`*`FILE`*
: imposta il nome del file di output su *FILE*

`--version`
: mostra informazioni sulla versione

### Problemi noti e avvertimenti

La lista degli script per gestire le articolazioni è incompleta. Le misure vuote confondono `etf2ly`. Le sequenze di abbellimenti non sono risolte correttamente.

### 4.3.5 Altri formati

LilyPond non supporta la conversione da altri formati, ma esistono alcuni strumenti esterni che possono generare file LilyPond. L'elenco si trova in Sezione "Easier editing" in *Informazioni generali*.

## 4.4 Inclusione di partiture LilyPond in altri programmi

Questa sezione presenta dei metodi per integrare testo e musica diversi dal metodo automatizzato di `lilypond-book`.

### Tante citazioni da una grande partitura

Per inserire molti frammenti di una grande partitura, si può usare anche la funzione di ritaglio dei sistemi; si veda Sezione "Extracting fragments of music" in *Guida alla Notazione*.

### Inserire l'output di LilyPond in OpenOffice.org

La notazione di LilyPond può essere aggiunta a OpenOffice.org con OOoLilyPond.

### Inserire l'output di LilyPond in altri programmi

Per inserire l'output di LilyPond in altri programmi, si usa `lilypond` invece di `lilypond-book`. Bisogna creare ogni esempio singolarmente e aggiungerlo al documento; consulta la documentazione del realtivo programma. La maggior parte dei programmi può inserire l'output di LilyPond in formato '`PNG`', '`EPS`' o '`PDF`'.

Per ridurre lo spazio bianco intorno alla partitura LilyPond, si usano le seguenti opzioni

```
\paper{
  indent=0\mm
  line-width=120\mm
  oddFooterMarkup=##f
  oddHeaderMarkup=##f
  bookTitleMarkup = ##f
  scoreTitleMarkup = ##f
}

{ c1 }
```

Per creare file immagine utili, si usa

```
EPS

lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts myfile.ly

PNG

lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts --png miofile.ly

Un file PNG transparente

lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts \
  -dpixmap-format=pngalpha --png miofile.ly
```

## 4.5 `include` indipendenti

Alcuni hanno scritto ampi (e utili!) frammenti di codice che possono essere condivisi da progetti diversi. Questo codice potrebbe alla fine entrare a far parte di LilyPond, ma finché questo non avviene occorre scaricarlo e includerlo manualmente con `\include`.

### 4.5.1 Articolazione MIDI

LilyPond permette di generare l'output MIDI, che consente di "verificare a orecchio" quanto è stato scritto. Tuttavia l'output contiene solo la dinamica, i segni di tempo espliciti, le note e le loro durate.

Il progetto *articulate* costituisce un tentativo di inviare al file MIDI maggiori informazioni sulla partitura. Il suo funzionamento si basa sull'abbreviazione delle note prive di legatura di portamento, in modo da 'articolare' le note. Questa riduzione dipende dai segni di articolazione attaccati a una nota: staccato dimezza il valore della nota, tenuto assegna alla nota la sua intera durata, e così via. Lo script è consapevole anche dei trilli e dei gruppetti; può essere esteso per elaborare altri ornamenti come i mordenti.

http://www.nicta.com.au/people/chubbp/articulate

#### Problemi noti e avvertimenti

La sua principale limitazione è che può agire solo sugli elementi che conosce: qualsiasi cosa che sia del semplice testo (invece di una proprietà della nota) viene dunque ignorato.

# 5 Consigli su come scrivere i file

Ora puoi iniziare a scrivere file di input di LilyPond più grandi – non più i piccoli esempi del tutorial, ma pezzi completi. Ma qual è il modo migliore di farlo?

Finché LilyPond comprende i file di input e produce l'output che desideri, non importa quale aspetto abbiano i file di input. Tuttavia, ci sono altri aspetti da tenere a mente quando si scrivono file di input di LilyPond.

- Che fare in caso di errore? La struttura data a un file LilyPond può rendere l'individuazione di certi tipi di errore più facile (o più difficile).
- Che fare se vuoi inviare i tuoi file di input a qualcuno? E se decidi di modificare i tuoi file di input dopo qualche anno? Alcuni file di input di LilyPond sono comprensibili a prima vista; altri ti possono lasciare a grattarti la testa per un'ora.
- Che fare se vuoi aggiornare il tuo file per poterlo usare con una versione più recente di LilyPond? Con l'evolversi di LilyPond, la sintassi di input si trova soggetta a occasionali cambiamenti. Alcune modifiche possono essere fatte in automatico con `convert-ly`, ma altre potrebbero richiedere un intervento manuale. I file di input di LilyPond possono essere strutturati in moda da essere poi aggiornati in modo più semplice (o più difficile).

## 5.1 Consigli generali

Ecco alcuni consigli che possono aiutarti a evitare o risolvere i problemi:

- **Includi il numero di `\version` in ogni file**. Nota che tutti i modelli contengono l'informazione su `\version`. Si consiglia vivamente di includere sempre `\version`, non importa quanto piccolo possa essere il file. L'esperienza personale insegna come sia frustrante cercare di ricordare quale versione di LilyPond si usava alcuni anni prima! `convert-ly` richiede che si dichiari la versione di LilyPond utilizzata.
- **Includi i controlli**: Sezione "Controlli di battuta e del numero di battuta" in *Guida alla Notazione*, Sezione "Controlli di ottava" in *Guida alla Notazione*. Includendo i controlli ogni tanto, se fai un errore lo puoi individuare più rapidamente. Cosa si intende per 'ogni tanto'? Dipende dalla complessità della musica. Se la musica è molto semplice, anche solo una volta o due. Se la musica è molto complessa, a ogni battuta.
- **Una battuta per ogni linea di testo**. Se c'è qualcosa di complicato, nella musica stessa o nell'output che desideri, di solito è preferibile scrivere una sola battuta per linea. Risparmiare spazio sullo schermo concentrando otto battute per ogni riga non è affatto conveniente se poi devi fare il 'debug' dei file di input.
- **Inserisci dei commenti nei file di input**. Puoi usare i numeri di battuta (ogni tanto) o dei riferimenti ai temi musicali ('secondo tema nei violini,' 'quarta variazione,' etc.). Potresti non aver bisogno dei commenti mentre scrivi il brano la prima volta, ma se due o tre anni dopo vuoi cambiare qualcosa o se vuoi dare il sorgente a un amico, sarà molto più difficile capire le tue intenzioni e la struttura del file se mancano i commenti.
- **Indenta le parentesi graffe**. Molti problemi sono causati da mancata corrispondenza tra le quantità di `{` e di `}`.
- **Esplicita le durate** all'inizio delle sezioni e delle variabili. Se specifichi `c4 d e` all'inizio di una frase (invece di `c d e` soltanto) puoi evitare l'insorgere di problemi al momento di rimetter mano alla musica.
- **Separa le modifiche manuali (tweak)** dalle definizioni musicali. Vedi Sezione "Ridurre l'input grazie a variabili e funzioni" in *Manuale di Apprendimento*, e Sezione "Fogli di stile" in *Manuale di Apprendimento*.

## 5.2 Scrivere musica esistente

Se stai riportando della musica da una partitura esistente (ovvero il brano contenuto in uno spartito già scritto),

- Inserisci in LilyPond le note del manoscritto (la copia fisica della musica) un sistema alla volta (ma sempre una battuta per linea di testo), e controlla ogni sistema completato. Puoi usare le proprietà `showLastLength` o `showFirstLength` per velocizzare l'elaborazione – vedi Sezione "Skipping corrected music" in *Guida alla Notazione*.
- Definisci `mBreak = { \break }` e inserisci `\mBreak` nel file di input ogni volta che nel manoscritto c'è un a capo. In questo modo è più semplice confrontare la musica generata da LilyPond con quella originale. Quando hai finito la revisione della partitura, puoi definire `mBreak = { }` per eliminare tutte queste interruzioni di riga. Così LilyPond potrà inserire le interruzioni dove ritiene stiano meglio.
- Quando si inserisce una parte per strumento traspositore all'interno di una variabile, è consigliabile racchiudere le note tra parentesi graffe

  `\transpose c altezza-naturale {...}`

  (dove `altezza-naturale` corrisponde all'intonazione di base dello strumento) così che la musica contenuta nella variabile sia effettivamente scritta in Do. La puoi presentare trasposta quando la variabile viene usata, se necessario, ma potresti non desiderarlo (ad esempio quando si stampa una partitura in intonazione reale, quando si traspone una parte per trombone dalla chiave di Sol alla chiave di basso, etc.). Errori nelle trasposizioni sono meno probabili se tutta la musica contenuta nelle variabili è ad un'altezza costante.

  Inoltre, trasponi sempre in relazione al Do. Questo significa che le uniche altre tonalità che userai saranno le altezze naturali degli strumenti - bes per una tromba in Si bemolle, aes per un clarinetto in La bemolle, etc.

## 5.3 Grandi progetti

Quando si lavora a un grande progetto, definire una struttura chiara nel file di input diventa vitale.

- **Usa una variabile per ogni voce**, con un minimo di struttura nella definizione. La struttura della sezione `\score` è la parte più probabilmente soggetta a cambiamenti; è invece molto improbabile che la definizione di `violin` cambi in una nuova versione di LilyPond.

```
violin = \relative c'' {
g4 c'8. e16
}
...
\score {
  \new GrandStaff {
    \new Staff {
      \violin
    }
  }
}
```

- **Separa le modifiche manuali (tweak) dalle definizioni musicali**. Questo punto è stato menzionato prima; nei grandi progetti diventa di vitale importanza. Potrebbe essere necessario modificare la definizione di `fthenp`, ma si dovrebbe farlo una volta sola e senza toccare niente in `violin`.

```
fthenp = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
```

```
    g4\fthenp c'8. e16
    }
```

## 5.4 Risoluzione dei problemi

Prima o poi ti capiterà di scrivere un file che LilyPond non riesce a compilare. I messaggi inviati da LilyPond potrebbero aiutarti a trovare l'errore, ma in molti casi sarà necessario fare qualche ricerca per individuare l'origine del problema.

Gli strumenti più potenti a questo riguardo sono il commento della linea singola (indicato da `%`) e il commento di blocco (indicato da `%{ ... %}`). Se non sai dove sia il problema, inizia col commentare ampie parti del file di input. Dopo aver commentato una sezione, prova a compilare di nuovo il file. Se funziona, allora il problema deve trovarsi nella parte che hai appena commentato. Se non funziona, continua a commentare il materiale finché non ottieni un codice funzionante.

Nel caso estremo, potresti finire con soltanto

```
\score {
  <<
    % \melody
    % \harmony
    % \bass
  >>
  \layout{}
}
```

(in altre parole, un file senza musica)

Se dovesse succedere, non rinunciare. Decommenta un pezzetto – ad esempio, la parte di basso – e vedi se funziona. Se non funziona, allora commenta tutta la musica del basso (ma lascia `\bass` in `\score` non commentato).

```
bass = \relative c' {
%{
  c4 c c c
  d d d d
%}
}
```

Ora inizia a decommentare mano a mano la parte di `bass` finché non trovi la linea che causa il problema.

Un'altra tecnica di debug molto utile consiste nel creare Sezione "Esempi minimi" in *Informazioni generali*.

## 5.5 Make e Makefile

Tutte le piattaforme su cui Lilypond può essere installato supportano un software chiamato `make`. Questo software legge un file speciale chiamato `Makefile` che definisce quali file dipendono da quali altri e quali comandi occorra dare al sistema operativo per produrre un file da un altro. Ad esempio Makefile può spiegare come generare '`ballad.pdf`' e '`ballad.midi`' da '`ballad.ly`' eseguendo Lilypond.

In alcune situazioni, è una buona idea creare un `Makefile` per il proprio progetto, per proprio comodo o come cortesia per quanti altri possano avere accesso ai file sorgente. Questo vale per i progetti molto ampi con tanti file inclusi e diverse opzioni di output (ad esempio, partitura completa, parti, partitura del direttore, riduzione per pianoforte, etc.) o per progetti che richiedono comandi difficili per la compilazione (come i progetti che usano `lilypond-book`). I Makefile variano molto in complessità e flessibilità, in base alle necessità e alle abilità degli

autori. Il programma GNU Make è installato nelle distribuzioni GNU/Linux e su MacOS X ed è disponibile anche per Windows.

Si veda il **Manuale di GNU Make** per conoscere in dettaglio l'uso di `make`, dato che quel che segue dà solo un'idea delle sue potenzialità.

I comandi per definire delle regole in un Makefile cambiano in base alla piattaforma; ad esempio le varie distribuzioni di GNU/Linux e MacOS usano `bash`, mentre Windows usa `cmd`. Nota che su MacOS X è necessario configurare il sistema per usare l'interprete da linea di comando. Di seguito alcuni Makefile di esempio, con versioni sia per GNU/Linux/MacOS sia per Windows.

Il primo esempio è per una composizione per orchestra in quattro movimenti e presenta una directory strutturata come segue:

```
Symphony/
|-- MIDI/
|-- Makefile
|-- Notes/
|   |-- cello.ily
|   |-- figures.ily
|   |-- horn.ily
|   |-- oboe.ily
|   |-- trioString.ily
|   |-- viola.ily
|   |-- violinOne.ily
|   `-- violinTwo.ily
|-- PDF/
|-- Parts/
|   |-- symphony-cello.ly
|   |-- symphony-horn.ly
|   |-- symphony-oboes.ly
|   |-- symphony-viola.ly
|   |-- symphony-violinOne.ly
|   `-- symphony-violinTwo.ly
|-- Scores/
|   |-- symphony.ly
|   |-- symphonyI.ly
|   |-- symphonyII.ly
|   |-- symphonyIII.ly
|   `-- symphonyIV.ly
`-- symphonyDefs.ily
```

I file '`.ly`' nelle directory '`Scores`' e '`Parts`' prendono le note dai file '`.ily`' nella directory '`Notes`':

```
%%% inizio del file "symphony-cello.ly"
\include ../symphonyDefs.ily
\include ../Notes/cello.ily
```

Il Makefile avrà i target di `score` (l'intero brano in partitura completa), `movements` (singoli movimenti in partitura completa), e `parts` (singole parti per i musicisti). C'è anche un target `archive` che creerà un archivio compresso dei file sorgenti, utile per la condivisione via web o email. Ecco un esempio di Makefile per GNU/Linux e MacOS X. Dovrebbe essere salvato col nome `Makefile` nella directory principale del progetto:

**Nota:** Quando si definisce un target o una regola di pattern, le linee successive devono iniziare con i tabulatori, non con gli spazi.

```
# Il prefisso al nome dei file di output
piece = symphony
# Determinazione del numero dei processori
CPU_CORES=`cat /proc/cpuinfo | grep -m1 "cpu cores" | sed s/".*: "//`
# Il comando per eseguire lilypond
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click -djob-count=$(CPU_CORES)

# I suffissi usati in questo Makefile.
.SUFFIXES: .ly .ily .pdf .midi

# I file di input e di output vengono cercati nelle directory elencate
# nella variabile VPATH.  Tutte queste sono sottodirectory della directory
# corrente (assegnata dalla variabile `CURDIR' di GNU make).
VPATH = \
  $(CURDIR)/Scores \
  $(CURDIR)/PDF \
  $(CURDIR)/Parts \
  $(CURDIR)/Notes

# La regola di pattern per creare i file PDF e MIDI da un file di input LY.
# I file di output .pdf vengono messi nella sottodirectory `PDF', mentre i file
# .midi vanno nella sottodirectory `MIDI'.
%.pdf %.midi: %.ly
        $(LILY_CMD) $<; \           # questa linea inizia con una tabulazione
        if test -f "$*.pdf"; then \
            mv "$*.pdf" PDF/; \
        fi; \
        if test -f "$*.midi"; then \
            mv "$*.midi" MIDI/; \
        fi

notes = \
  cello.ily \
  horn.ily \
  oboe.ily \
  viola.ily \
  violinOne.ily \
  violinTwo.ily

# Le dipendenze dei movimenti.
$(piece)I.pdf: $(piece)I.ly $(notes)
$(piece)II.pdf: $(piece)II.ly $(notes)
$(piece)III.pdf: $(piece)III.ly $(notes)
$(piece)IV.pdf: $(piece)IV.ly $(notes)

# Le dipendenze della partitura completa.
$(piece).pdf: $(piece).ly $(notes)

# Le dipendenze delle parti.
```

```
$(piece)-cello.pdf: $(piece)-cello.ly cello.ily
$(piece)-horn.pdf: $(piece)-horn.ly horn.ily
$(piece)-oboes.pdf: $(piece)-oboes.ly oboe.ily
$(piece)-viola.pdf: $(piece)-viola.ly viola.ily
$(piece)-violinOne.pdf: $(piece)-violinOne.ly violinOne.ily
$(piece)-violinTwo.pdf: $(piece)-violinTwo.ly violinTwo.ily

# Lanciare `make score' per generare la partitura completa di tutti i quattro
# movimenti in un unico file.
.PHONY: score
score: $(piece).pdf

# Lanciare `make parts' per generare tutte le parti.
# Lanciare `make foo.pdf' per generare la parte per lo strumento `foo'.
# Esempio: `make symphony-cello.pdf'.
.PHONY: parts
parts: $(piece)-cello.pdf \
       $(piece)-violinOne.pdf \
       $(piece)-violinTwo.pdf \
       $(piece)-viola.pdf \
       $(piece)-oboes.pdf \
       $(piece)-horn.pdf

# Lanciare `make movements' per generare i file per i
# quattro movimenti separatamente.
.PHONY: movements
movements: $(piece)I.pdf \
           $(piece)II.pdf \
           $(piece)III.pdf \
           $(piece)IV.pdf

all: score parts movements

archive:
        tar -cvvf stamitz.tar \       # questa linea inizia con una tabulazione
        --exclude=*pdf --exclude=*~ \
        --exclude=*midi --exclude=*.tar \
        ../Stamitz/*
```

Ci sono alcune complicazioni specifiche della piattaforma Windows. Dopo aver scaricato e installato GNU Make per Windows, bisogna impostare il percorso corretto nelle variabili d'ambiente di sistema perché la shell DOS possa trovare il programma Make. Per farlo, clicca col tasto destro del mouse su "My Computer," poi scegli `Proprietà` e `Avanzate`. Clicca su `Variabili di ambiente`, e poi nel pannello `Variabili di Sistema`, nella sezione `Percorso`, clicca su `modifica` e aggiungi il percorso al file eseguibile GNU Make, che avrà un aspetto simile:

```
C:\Program Files\GnuWin32\bin
```

Lo stesso Makefile deve essere modificato per gestire diversi comandi shell e gli spazi che sono presenti in alcune directory predefinite di sistema. Il target `archive` target viene tolto perché Windows non ha il comando `tar`; inoltre Windows ha una diversa estensione predefinita per i file midi.

```
## VERSIONE DI WINDOWS
```

```
##
piece = symphony
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click \
                    -djob-count=$(NUMBER_OF_PROCESSORS)

#get the 8.3 name of CURDIR (workaround for spaces in PATH)
workdir = $(shell for /f "tokens=*" %%b in ("$(CURDIR)") \
          do @echo %%~sb)

.SUFFIXES: .ly .ily .pdf .mid

VPATH = \
  $(workdir)/Scores \
  $(workdir)/PDF \
  $(workdir)/Parts \
  $(workdir)/Notes

%.pdf %.mid: %.ly
        $(LILY_CMD) $<      # questa linea inizia con una tabulazione
        if exist "$*.pdf"  move /Y "$*.pdf"  PDF/
        if exist "$*.mid" move /Y "$*.mid" MIDI/

notes = \
  cello.ily \
  figures.ily \
  horn.ily \
  oboe.ily \
  trioString.ily \
  viola.ily \
  violinOne.ily \
  violinTwo.ily

$(piece)I.pdf: $(piece)I.ly $(notes)
$(piece)II.pdf: $(piece)II.ly $(notes)
$(piece)III.pdf: $(piece)III.ly $(notes)
$(piece)IV.pdf: $(piece)IV.ly $(notes)

$(piece).pdf: $(piece).ly $(notes)

$(piece)-cello.pdf: $(piece)-cello.ly cello.ily
$(piece)-horn.pdf: $(piece)-horn.ly horn.ily
$(piece)-oboes.pdf: $(piece)-oboes.ly oboe.ily
$(piece)-viola.pdf: $(piece)-viola.ly viola.ily
$(piece)-violinOne.pdf: $(piece)-violinOne.ly violinOne.ily
$(piece)-violinTwo.pdf: $(piece)-violinTwo.ly violinTwo.ily

.PHONY: score
score: $(piece).pdf

.PHONY: parts
parts: $(piece)-cello.pdf \
```

```
       $(piece)-violinOne.pdf \
       $(piece)-violinTwo.pdf \
       $(piece)-viola.pdf \
       $(piece)-oboes.pdf \
       $(piece)-horn.pdf

.PHONY: movements
movements: $(piece)I.pdf \
           $(piece)II.pdf \
           $(piece)III.pdf \
           $(piece)IV.pdf

all: score parts movements
```

Il Makefile seguente è per un documento `lilypond-book` fatto con LaTeX. Questo progetto ha un indice, dunque il comando `latex` deve essere eseguito due volte per aggiornare i collegamenti. I file di output sono tutti salvati nella directory `out` per i file .pdf e nella directory `htmlout` per i file html.

```
SHELL=/bin/sh
FILE=myproject
OUTDIR=out
WEBDIR=htmlout
VIEWER=acroread
BROWSER=firefox
LILYBOOK_PDF=lilypond-book --output=$(OUTDIR) --pdf $(FILE).lytex
LILYBOOK_HTML=lilypond-book --output=$(WEBDIR) $(FILE).lytex
PDF=cd $(OUTDIR) && pdflatex $(FILE)
HTML=cd $(WEBDIR) && latex2html $(FILE)
INDEX=cd $(OUTDIR) && makeindex $(FILE)
PREVIEW=$(VIEWER) $(OUTDIR)/$(FILE).pdf &

all: pdf web keep

pdf:
        $(LILYBOOK_PDF)  # inizia con una tabulazione
        $(PDF)           # inizia con una tabulazione
        $(INDEX)         # inizia con una tabulazione
        $(PDF)           # inizia con una tabulazione
        $(PREVIEW)       # inizia con una tabulazione

web:
        $(LILYBOOK_HTML) # inizia con una tabulazione
        $(HTML)          # inizia con una tabulazione
        cp -R $(WEBDIR)/$(FILE)/ ./  # inizia con una tabulazione
        $(BROWSER) $(FILE)/$(FILE).html &  # inizia con una tabulazione

keep: pdf
        cp $(OUTDIR)/$(FILE).pdf $(FILE).pdf  # inizia con una tabulazione

clean:
        rm -rf $(OUTDIR) # inizia con una tabulazione
```

```
web-clean:
        rm -rf $(WEBDIR) # inizia con una tabulazione

archive:
        tar -cvvf myproject.tar \ # inizia questa linea con una tabulazione
        --exclude=out/* \
        --exclude=htmlout/* \
        --exclude=myproject/* \
        --exclude=*midi \
        --exclude=*pdf \
        --exclude=*~ \
        ../MyProject/*
```

Il Makefile precedente non funziona su Windows. Un'alternativa per gli utenti Windows consiste nel creare un semplice file batch contenente i comandi per la compilazione. Questo file non terrà traccia delle dipendenze come fa invece un Makefile, ma almeno riduce il processo di compilazione a un solo comando. Salva il codice seguente come `build.bat` o `build.cmd`. Il file batch può essere eseguito nel prompt DOS o semplicemente con un doppio clic sulla sua icona.

```
lilypond-book --output=out --pdf myproject.lytex
cd out
pdflatex myproject
makeindex myproject
pdflatex myproject
cd ..
copy out\myproject.pdf MyProject.pdf
```

## Vedi anche

Questo manuale: Sezione 1.2 [Uso da linea di comando], pagina 1, Capitolo 3 [lilypond-book], pagina 19

# Appendice A 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 B Indice di LilyPond


## \

\header nei documenti LaTeX ..... 24

## A

ABC ..... 44
aggiornare i vecchi file di input ..... 15
Aggiornare un file di LilyPond ..... 15
avvertimento ..... 10

## C

caratteri vettoriali ..... 30
Coda Technology ..... 44
colorazione della sintassi ..... 40
convert-ly ..... 15

## D

dimensione del file di output ..... 39
directory, dirigere l'output in ..... 3
docbook ..... 19
DocBook, aggiungere musica ..... 19
documenti, aggiungere musica ..... 19
dvips ..... 30

## E

`Easier editing` ..... 45
`Editing facilitato` ..... 41
editor ..... 40
emacs ..... 40
enigma ..... 44
errore ..... 10
Errore di programmazione ..... 11
errore fatale ..... 10
errore Scheme ..... 11
errori, formato del messaggio ..... 11
`Esempi minimi` ..... 49
ETF ..... 44
Evince ..... 39

## F

file di output, dimensione ..... 39
Finale ..... 44
formato, output ..... 2

## G

gabbia chroot, esecuzione all'interno di ..... 2

## H

HTML ..... 19
HTML, aggiungere musica ..... 19

## I

immagine di anteprima ..... 26

## L

LANG ..... 8
LaTex ..... 19
LaTeX, aggiungere musica ..... 19
LILYPOND_DATADIR ..... 8
linea di comando, opzioni di ..... 2
loglevel ..... 3

## M

`MacOS X` ..... 1, 19, 41
make ..... 49
makefile ..... 49
`Manuali` ..... 1
messaggi di errore ..... 10
MIDI ..... 41
miniatura ..... 26
modalità, editor ..... 40
musicologia ..... 19
MusicXML ..... 43

## O

OpenOffice.org ..... 45
opzioni della linea di comando per `lilypond` ..... 2
output dettagliato ..... 3
output, directory ..... 3
output, formato ..... 2
output, impostare il nome del file ..... 3
output, PDF (Portable Document Format) ..... 4
output, PNG (Portable Network Graphics) ..... 4
output, PS (Postscript) ..... 4

## P

PDF (Portable Document Format), output ..... 4
percorso di ricerca ..... 2
PNG (Portable Network Graphics), output ..... 4
Postscript (PS), output ..... 4
Programmi esterni, generare file LilyPond ..... 45
PS (Postscript), output ..... 4
punta e clicca ..... 38
punta e clicca, linea di comando ..... 4

## R

ricerca dei file ..... 2

## S

Scheme, valutazione dell'espressione ..... 2
sintassi, colorazione ..... 40
Sospeso (core dumped) ..... 11
switch ..... 2

## T

texi ..... 19
texinfo ..... 19
Texinfo, aggiungere musica ..... 19
titoli e lilypond-book ..... 24
titoli in HTML ..... 26
traccia di chiamata ..... 11
traccia, Scheme ..... 11
type1, carattere ..... 30

## U

utilizzo di dvips ..... 30
Utilizzo di `lilypond` ..... 2

## V

valutazione dell'espressione, Scheme ..... 2
vim ..... 40

## W

`Windows` ..... 19

## X

Xpdf ..... 38