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 2.14.1. 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 "Manuals" in *Informazioni generali*.

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

Copyright © 1999–2011 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 2.14.1

Sommario

1 Eseguire `lilypond` ... 1

1.1 Uso normale ... 1
1.2 Uso da linea di comando ... 1
Invocare `lilypond` ... 1
Comandi standard da shell ... 1
Opzioni della linea di comando per `lilypond` ... 2
Variabili d'ambiente ... 5
LilyPond in una gabbia chroot ... 5
1.3 Messaggi di errore ... 7
1.4 Errori comuni ... 8
La musica esce dalla pagina ... 8
Appare un rigo in più ... 8
Errore apparente in `../ly/init.ly` ... 9
Messaggio di errore Unbound variable % ... 10
Messaggio di errore FT_Get_Glyph_Name ... 10
Avvertimento sul fatto che le affinità del rigo devono solo diminuire ... 10

2 Aggiornare i file con `convert-ly` ... 11

2.1 Perché la sintassi cambia? ... 11
2.2 Invocare `convert-ly` ... 11
2.3 Opzioni da linea di comando per `convert-ly` ... 12
2.4 Problemi nell'eseguire `convert-ly` ... 12
2.5 Conversioni manuali ... 12

3 Running `lilypond-book` ... 14

3.1 An example of a musicological document ... 14
3.2 Integrating music and text ... 18
3.2.1 LaTeX ... 18
3.2.2 Texinfo ... 19
3.2.3 HTML ... 20
3.2.4 DocBook ... 21
3.3 Music fragment options ... 21
3.4 Invoking `lilypond-book` ... 24
3.5 Filename extensions ... 26
3.6 lilypond-book templates ... 27
3.6.1 LaTeX ... 27
3.6.2 Texinfo ... 27
3.6.3 html ... 28
3.6.4 xelatex ... 28
3.7 Sharing the table of contents ... 29
3.8 Alternative methods of mixing text and music ... 31

4 External programs ... 32

4.1 Point and click ... 32
4.2 Text editor support ... 33
Emacs mode ... 33
Vim mode ... 33

Other editors ... 33
4.3 Converting from other formats ... 33
4.3.1 Invoking `midi2ly` ... 33
4.3.2 Invoking `musicxml2ly` ... 35
4.3.3 Invoking `abc2ly` ... 35
4.3.4 Invoking `etf2ly` ... 36
4.3.5 Other formats ... 37
4.4 LilyPond output in other programs ... 37
Many quotes from a large score ... 37
Inserting LilyPond output into OpenOffice.org ... 37
Inserting LilyPond output into other programs ... 37
4.5 Independent `includes` ... 37
4.5.1 MIDI articulation ... 37

5 Consigli su come scrivere i file ... 39
5.1 Consigli generali ... 39
5.2 Scrivere musica esistente ... 40
5.3 Grandi progetti ... 40
5.4 Risoluzione dei problemi ... 41
5.5 Make e Makefiles ... 41

Appendice A GNU Free Documentation License ... 48

Appendice B Indice di LilyPond ... 55

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.

Invocare 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 della linea di comando per lilypond

Sono contemplate le seguenti opzioni:

`-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`.

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

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

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

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

Imposta l'opzione interna del programma, *variabile*, al valore di Scheme *valore*. Se *valore* non viene specificato, allora viene usato *#t*. Per disabilitare un'opzione, si può usare il prefisso `no-` prima della *variabile*, ad esempio

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

è equivalente a

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

Di seguito alcune opzioni interessanti.

'`help`' L'esecuzione di `lilypond -dhelp` mostrerà tutte le opzioni disponibili di `-d`.

'`paper-size`'

Questa opzione imposta la dimensione predefinita del foglio,

```
-dpaper-size=\"letter\"
```

Nota che la stringa deve essere compresa tra virgolette precedute dal segno di escape (`\"`).

'`safe`' 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:export (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.

'`backend`' il formato di output da usare per il back-end. Per il `formato` si può scegliere tra

`ps` per PostScript.

I file Postscript 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`

per PostScript incapsulato. 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`.

`svg`

per ottenere 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.

`scm`

per estrarre i comandi di disegno grezzi e interni, basati su Scheme.

`null` non produce la stampa della partitura; ha lo stesso effetto di `-dno-print-pages`.

Esempio: `lilypond -dbackend=svg` *`filename`*`.ly`

'`preview`' Genera un file di output che contiene i titoli e il primo sistema del brano musicale. Se si usano i blocchi `\bookpart`, i titoli e il primo sistema di ogni `\bookpart` apparirà nell'output. I backend `ps`, `eps`, e `svg` supportano questa opzione.

'`print-pages`'
Genera tutte le pagine, come da impostazione predefinita. `-dno-print-pages` è utile in combinazione con `-dpreview`.

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

`-H,--header=`*`CAMPO`*
: Estrae un campo dell'intestazione nel file '`NOME.`*`CAMPO`*'.

`--include, -I=`*`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.

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

`-o,--output=`*`FILE`* or *`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`.

`-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.

  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 installation (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.

`-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**!)

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_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 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 8.

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.
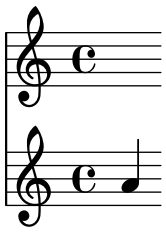
La durata errata può essere trovata facilmente se si usano i controlli di battuta, si veda Sezione "Bar and bar number checks" in *Guida alla Notazione*.

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 "Bar lines" 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 }
```
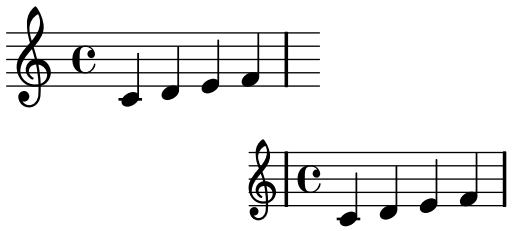


Questo accade perché non esiste un contesto `Staff` quando viene elaborata l'istruzione di override, quindi ne viene implicitamente creato uno e l'override viene applicato ad esso. Ma poi il comando `\new Staff` crea un altro rigo separato nel quale vengono inserite le note. Il codice corretto per colorare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

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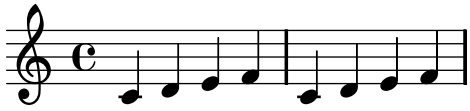


Vediamo un secondo esempio. Se un comando `\relative` viene posto dentro un comando `\repeat`, vengono generati due righi, il secondo spostato orizzontalmente rispetto al primo, perché il comando `\repeat` genera due blocchi `\relative`, ognuno dei quali crea implicitamente i blocchi `Staff` e `Voice`.

```
\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 }
  }
}
```



Errore apparente in `../ly/init.ly`

Possono apparire diversi strani messaggi di errore relativi a errori di sintassi in '`../ly/init.ly`' se il file di input non ha una forma corretta, ad esempio se contiene delle parentesi o delle virgolette non chiuse correttamente.

L'errore più comune è la mancanza di una parentesi graffa, (`}`), alla fine di un blocco `score`. In questo caso la soluzione è ovvia: controlla che il blocco `score` sia chiuso correttamente. La struttura corretta di un file di input è descritta in Sezione "Come funzionano i file di input di

LilyPond" in *Manuale di Apprendimento*. Per evitare questi errori conviene usare un editor che evidenzi automaticamente le parentesi e le graffe corrispondenti.

Un'altra causa frequente di errore è la mancanza di uno spazio tra l'ultima sillaba di un blocco di testo (lyrics) e la parentesi graffa che chiude il blocco, (`}`). Senza questa separazione, la graffa viene considerata come parte della sillaba. Si consiglia di assicurarsi sempre che ci sia uno spazio prima e dopo *ogni* parentesi graffa. Per comprendere l'importanza di questo quando si usa il testo, si veda Sezione "Entering lyrics" in *Guida alla Notazione*.

Questo messaggio di errore può apparire anche nel caso in cui sia omessa la virgoletta di chiusura, (`"`). In questo caso il messaggio di errore dovrebbe dare un numero di riga vicino alla riga sbagliata. La virgoletta non chiusa sarà solitamente una o due righe sopra.

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*.

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 Invocare 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:

`-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`

`-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.

`--to=to-patchlevel`
: Imposta la versione obiettivo della conversione. L'impostazione predefinita è l'ultima versione disponibile. Esempio: `--to=2.12.2`

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

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].

[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.

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

3 Running lilypond-book

If you want to add pictures of music to a document, you can simply do it the way you would do with other types of pictures. The pictures are created separately, yielding PostScript output or PNG images, and those are included into a LaTeX or HTML document.

`lilypond-book` provides a way to automate this process: This program extracts snippets of music from your document, runs `lilypond` on them, and outputs the document with pictures substituted for the music. The line width and font size definitions for the music are adjusted to match the layout of your document.

This is a separate program from `lilypond` itself, and is run on the command line; for more information, see ⟨undefined⟩ [Command-line usage], pagina ⟨undefined⟩. If you have MacOS 10.3 or 10.4 and you have trouble running `lilypond-book`, see Sezione "MacOS X" in *Informazioni generali*.

This procedure may be applied to LaTeX, HTML, Texinfo or DocBook documents.

3.1 An example of a musicological document

Some texts contain music examples. These texts are musicological treatises, songbooks, or manuals like this. Such texts can be made by hand, simply by importing a PostScript figure into the word processor. However, there is an automated procedure to reduce the amount of work involved in HTML, LaTeX, Texinfo and DocBook documents.

A script called `lilypond-book` will extract the music fragments, format them, and put back the resulting notation. Here we show a small example for use with LaTeX. The example also contains explanatory text, so we will not comment on it further.

Input

```
\documentclass[a4paper]{article}

\begin{document}

Documents for \verb+lilypond-book+ may freely mix music and text.
For example,

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

Options are put in brackets.

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

Larger examples can be put into a separate file, and introduced with
\verb+\lilypondfile+.

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

(If needed, replace @file{screech-boink.ly} by any @file{.ly} file
```

```
you put in the same directory as this file.)

\end{document}
```

Processing

Save the code above to a file called ‘lilybook.lytex’, then in a terminal run

```
lilypond-book --output=out --pdf lilybook.lytex
lilypond-book (GNU LilyPond) 2.14.1

Reading lilybook.lytex...
..lots of stuff deleted..
Compiling lilybook.tex...
cd out
pdflatex lilybook
..lots of stuff deleted..
xpdf lilybook
(replace xpdf by your favorite PDF viewer)
```
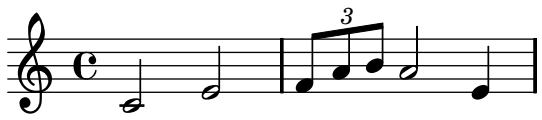
Running `lilypond-book` and `latex` creates a lot of temporary files, which would clutter up the working directory. To remedy this, use the `--output=dir` option. It will create the files in a separate subdirectory ‘dir’.

Finally the result of the LaTeX example shown above.[1] This finishes the tutorial section.

[1] This tutorial is processed with Texinfo, so the example gives slightly different results in layout.
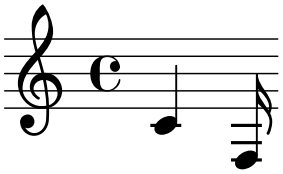
Output

Documents for `lilypond-book` may freely mix music and text. For example,

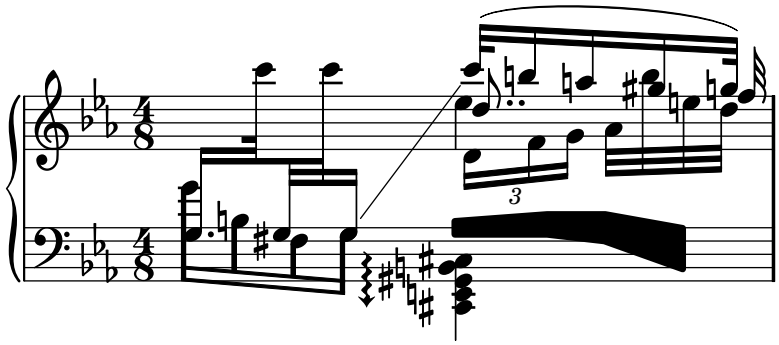


Options are put in brackets.

```
c'4 f16
```



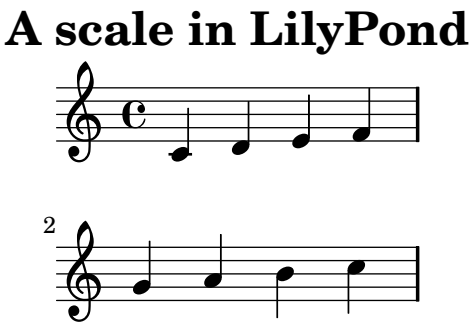
Larger examples can be put into a separate file, and introduced with `\lilypondfile`.



If a `tagline` is required, either default or custom, then the entire snippet must be enclosed in a `\book { }` construct.

```
\book{
  \header{
    title = "A scale in LilyPond"
  }

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



Music engraving by LilyPond 2.14.1—www.lilypond.org

3.2 Integrating music and text

Here we explain how to integrate LilyPond with various output formats.

3.2.1 LaTeX

LaTeX is the de-facto standard for publishing layouts in the exact sciences. It is built on top of the TeX typesetting engine, providing the best typography available anywhere.

See *The Not So Short Introduction to LaTeX* for an overview on how to use LaTeX.

Music is entered using

```
\begin{lilypond}[options,go,here]
  YOUR LILYPOND CODE
\end{lilypond}
```

or

```
\lilypondfile[options,go,here]{filename}
```

or

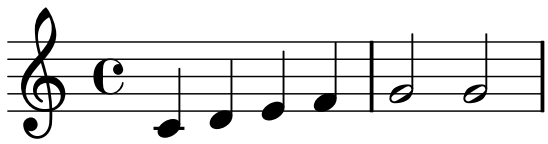
```
\lilypond[options,go,here]{ YOUR LILYPOND CODE }
```

Additionally, `\lilypondversion` displays the current version of lilypond. Running `lilypond-book` yields a file that can be further processed with LaTeX.

We show some examples here. The `lilypond` environment

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

produces



The short version

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

produces



Currently, you cannot include `{` or `}` within `\lilypond{}`, so this command is only useful with the `fragment` option.

The default line width of the music will be adjusted by examining the commands in the document preamble, the part of the document before `\begin{document}`. The `lilypond-book` command sends these to LaTeX to find out how wide the text is. The line width for the music fragments is then adjusted to the text width. Note that this heuristic algorithm can fail easily; in such cases it is necessary to use the `line-width` music fragment option.

Each snippet will call the following macros if they have been defined by the user:

- `\preLilyPondExample` called before the music,
- `\postLilyPondExample` called after the music,
- `\betweenLilyPondSystem[1]` is called between systems if `lilypond-book` has split the snippet into several PostScript files. It must be defined as taking one parameter and will be passed the number of files already included in this snippet. The default is to simply insert a `\linebreak`.

Frammenti di codice selezionati

Sometimes it is useful to display music elements (such as ties and slurs) as if they continued after the end of the fragment. This can be done by breaking the staff and suppressing inclusion of the rest of the LilyPond output.

In LaTeX, define `\betweenLilyPondSystem` in such a way that inclusion of other systems is terminated once the required number of systems are included. Since `\betweenLilyPondSystem` is first called *after* the first system, including only the first system is trivial.

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

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

If a greater number of systems is requested, a TeX conditional must be used before the `\endinput`. In this example, replace '2' by the number of systems you want in the output.

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

(Since `\endinput` immediately stops the processing of the current input file we need `\expandafter` to delay the call of `\endinput` after executing `\fi` so that the `\if-\fi` clause is balanced.)

Remember that the definition of `\betweenLilyPondSystem` is effective until TeX quits the current group (such as the LaTeX environment) or is overridden by another definition (which is, in most cases, for the rest of the document). To reset your definition, write

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

in your LaTeX source.

This may be simplified by defining a TeX macro

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

and then saying only how many systems you want before each fragment,

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

Vedi anche

There are specific `lilypond-book` command line options and other details to know when processing LaTeX documents, see Sezione 3.4 [Invoking lilypond-book], pagina 24.

3.2.2 Texinfo

Texinfo is the standard format for documentation of the GNU project. An example of a Texinfo document is this manual. The HTML, PDF, and Info versions of the manual are made from the Texinfo document.

In the input file, music is specified with

```
@lilypond[options,go,here]
  YOUR LILYPOND CODE
@end lilypond
```

or

```
@lilypond[options,go,here]{ YOUR LILYPOND CODE }
```

or

```
@lilypondfile[options,go,here]{filename}
```
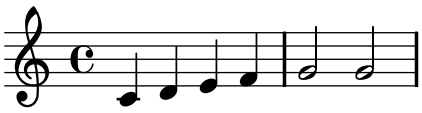
Additionally, `@lilypondversion` displays the current version of lilypond.

When `lilypond-book` is run on it, this results in a Texinfo file (with extension '`.texi`') containing `@image` tags for HTML, Info and printed output. `lilypond-book` generates images of the music in EPS and PDF formats for use in the printed output, and in PNG format for use in HTML and Info output.

We show two simple examples here. A `lilypond` environment

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

produces



The short version

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

produces
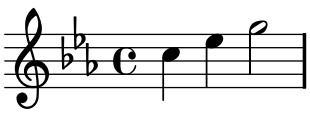


Contrary to LaTeX, `@lilypond{...}` does not generate an in-line image. It always gets a paragraph of its own.

3.2.3 HTML

Music is entered using

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

`lilypond-book` then produces an HTML file with appropriate image tags for the music fragments:



For inline pictures, use `<lilypond ... />`, where the options are separated by a colon from the music, for example

```
Some music in <lilypond relative=2: a b c/> a line of text.
```

To include separate files, say

```
<lilypondfile option1 option2 ...>filename</lilypondfile>
```

For a list of options to use with the `lilypond` or `lilypondfile` tags, see Sezione 3.3 [Music fragment options], pagina 21.

Additionally, `<lilypondversion/>` displays the current version of lilypond.

3.2.4 DocBook

For inserting LilyPond snippets it is good to keep the conformity of our DocBook document, thus allowing us to use DocBook editors, validation etc. So we don't use custom tags, only specify a convention based on the standard DocBook elements.

Common conventions

For inserting all type of snippets we use the `mediaobject` and `inlinemediaobject` element, so our snippets can be formatted inline or not inline. The snippet formatting options are always provided in the `role` property of the innermost element (see in next sections). Tags are chosen to allow DocBook editors format the content gracefully. The DocBook files to be processed with `lilypond-book` should have the extension '`.lyxml`'.

Including a LilyPond file

This is the most simple case. We must use the '`.ly`' extension for the included file, and insert it as a standard `imageobject`, with the following structure:

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

Note that you can use `mediaobject` or `inlinemediaobject` as the outermost element as you wish.

Including LilyPond code

Including LilyPond code is possible by using a `programlisting`, where the language is set to `lilypond` with the following structure:

```
<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>
```

As you can see, the outermost element is a `mediaobject` or `inlinemediaobject`, and there is a `textobject` containing the `programlisting` inside.

Processing the DocBook document

Running `lilypond-book` on our '`.lyxml`' file will create a valid DocBook document to be further processed with '`.xml`' extension. If you use dblatex, it will create a PDF file from this document automatically. For HTML (HTML Help, JavaHelp etc.) generation you can use the official DocBook XSL stylesheets, however, it is possible that you have to make some customization for it.

3.3 Music fragment options

In the following, a 'LilyPond command' refers to any command described in the previous sections which is handled by `lilypond-book` to produce a music snippet. For simplicity, LilyPond commands are only shown in LaTeX syntax.

Note that the option string is parsed from left to right; if an option occurs multiple times, the last one is taken.

The following options are available for LilyPond commands:

`staffsize=`*ht*
: Set staff size to *ht*, which is measured in points.

`ragged-right`
: Produce ragged-right lines with natural spacing, i.e., `ragged-right = ##t` is added to the LilyPond snippet. This is the default for the `\lilypond{}` command if no `line-width` option is present. It is also the default for the `lilypond` environment if the `fragment` option is set, and no line width is explicitly specified.

`noragged-right`
: For single-line snippets, allow the staff length to be stretched to equal that of the line width, i.e., `ragged-right = ##f` is added to the LilyPond snippet.

`line-width`
`line-width=`*size*`\`*unit*
: Set line width to *size*, using *unit* as units. *unit* is one of the following strings: `cm`, `mm`, `in`, or `pt`. This option affects LilyPond output (this is, the staff length of the music snippet), not the text layout.

  If used without an argument, set line width to a default value (as computed with a heuristic algorithm).

  If no `line-width` option is given, `lilypond-book` tries to guess a default for `lilypond` environments which don't use the `ragged-right` option.

`papersize=`*string*
: Where *string* is a paper size defined in '`scm/paper.scm`' i.e. `a5`, `quarto`, `11x17` etc.

  Values not defined in '`scm/paper.scm`' will be ignored, a warning will be posted and the snippet will be printed using the default `a4` size.

`notime` Do not print the time signature, and turns off the timing (time signature, bar lines) in the score.

`fragment` Make `lilypond-book` add some boilerplate code so that you can simply enter, say,

  ```
  c'4
  ```

  without `\layout`, `\score`, etc.

`nofragment`
: Do not add additional code to complete LilyPond code in music snippets. Since this is the default, `nofragment` is redundant normally.

`indent=`*size*`\`*unit*
: Set indentation of the first music system to *size*, using *unit* as units. *unit* is one of the following strings: `cm`, `mm`, `in`, or `pt`. This option affects LilyPond, not the text layout.

`noindent` Set indentation of the first music system to zero. This option affects LilyPond, not the text layout. Since no indentation is the default, `noindent` is redundant normally.

`quote` Reduce line length of a music snippet by $2*0.4$ in and put the output into a quotation block. The value '0.4 in' can be controlled with the `exampleindent` option.

`exampleindent`
: Set the amount by which the `quote` option indents a music snippet.

`relative`
`relative=n`
: Use relative octave mode. By default, notes are specified relative to middle C. The optional integer argument specifies the octave of the starting note, where the default `1` is middle C. `relative` option only works when `fragment` option is set, so `fragment` is automatically implied by `relative`, regardless of the presence of any `(no)fragment` option in the source.

LilyPond also uses `lilypond-book` to produce its own documentation. To do that, some more obscure music fragment options are available.

`verbatim`
: The argument of a LilyPond command is copied to the output file and enclosed in a verbatim block, followed by any text given with the `intertext` option (not implemented yet); then the actual music is displayed. This option does not work well with `\lilypond{}` if it is part of a paragraph.

  If `verbatim` is used in a `lilypondfile` command, it is possible to enclose verbatim only a part of the source file. If the source file contain a comment containing '`begin verbatim`' (without quotes), quoting the source in the verbatim block will start after the last occurrence of such a comment; similarly, quoting the source verbatim will stop just before the first occurrence of a comment containing '`end verbatim`', if there is any. In the following source file example, the music will be interpreted in relative mode, but the verbatim quote will not show the `relative` block, i.e.

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

  will be printed with a verbatim block like

```
  c4 e2 g4
  f2 e
```

  If you would like to translate comments and variable names in verbatim output but not in the sources, you may set the environment variable `LYDOC_LOCALEDIR` to a directory path; the directory should contain a tree of '`.mo`' message catalogs with `lilypond-doc` as a domain.

`addversion`
: (Only for Texinfo output.) Prepend line `\version @w{"@version{}"}` to `verbatim` output.

`texidoc`
: (Only for Texinfo output.) If `lilypond` is called with the '`--header=texidoc`' option, and the file to be processed is called '`foo.ly`', it creates a file '`foo.texidoc`' if there is a `texidoc` field in the `\header`. The `texidoc` option makes `lilypond-book` include such files, adding its contents as a documentation block right before the music snippet.

  Assuming the file '`foo.ly`' contains

```
\header {
  texidoc = "This file demonstrates a single note."
}
{ c'4 }
```

  and we have this in our Texinfo document '`test.texinfo`'

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

  the following command line gives the expected result

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

Most LilyPond test documents (in the '`input`' directory of the distribution) are small '`.ly`' files which look exactly like this.

For localization purpose, if the Texinfo document contains `@documentlanguage` *`LANG`* and '`foo.ly`' header contains a `texidoc`*`LANG`* field, and if `lilypond` is called with '`--header=texidoc`*`LANG`*', then '`foo.texidoc`*`LANG`*' will be included instead of '`foo.texidoc`'.

`lilyquote`

(Only for Texinfo output.) This option is similar to quote, but only the music snippet (and the optional verbatim block implied by `verbatim` option) is put into a quotation block. This option is useful if you want to `quote` the music snippet but not the `texidoc` documentation block.

`doctitle` (Only for Texinfo output.) This option works similarly to `texidoc` option: if `lilypond` is called with the '`--header=doctitle`' option, and the file to be processed is called '`foo.ly`' and contains a `doctitle` field in the `\header`, it creates a file '`foo.doctitle`'. When `doctitle` option is used, the contents of '`foo.doctitle`', which should be a single line of *text*, is inserted in the Texinfo document as `@lydoctitle` *`text`*. `@lydoctitle` should be a macro defined in the Texinfo document. The same remark about `texidoc` processing with localized languages also applies to `doctitle`.

`nogettext`

(Only for Texinfo output.) Do not translate comments and variable names in the snippet quoted verbatim.

`printfilename`

If a LilyPond input file is included with `\lilypondfile`, print the file name right before the music snippet. For HTML output, this is a link. Only the base name of the file is printed, i.e. the directory part of the file path is stripped.

3.4 Invoking `lilypond-book`

`lilypond-book` produces a file with one of the following extensions: '`.tex`', '`.texi`', '`.html`' or '`.xml`', depending on the output format. All of '`.tex`', '`.texi`' and '`.xml`' files need further processing.

Format-specific instructions

LaTeX

There are two ways of processing your LaTeX document for printing or publishing: getting a PDF file directly with PDFLaTeX, or getting a PostScript file with LaTeX via a DVI to PostScript translator like `dvips`. The first way is simpler and recommended[1], and whichever way you use, you can easily convert between PostScript and PDF with tools, like `ps2pdf` and `pdf2ps` included in Ghostscript package.

To produce a PDF file through PDFLaTeX, use

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

To produce PDF output via LaTeX/`dvips`/`ps2pdf`, you should do

[1] Note that PDFLaTeX and LaTeX may not be both usable to compile any LaTeX document, that is why we explain the two ways.

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

The '.dvi' file created by this process will not contain note heads. This is normal; if you follow the instructions, they will be included in the '.ps' and '.pdf' files.

Running dvips may produce some warnings about fonts; these are harmless and may be ignored. If you are running latex in twocolumn mode, remember to add -t landscape to the dvips options.

Texinfo

To produce a Texinfo document (in any output format), follow the normal procedures for Texinfo; this is, either call texi2pdf or texi2dvi or makeinfo, depending on the output format you want to create. See the documentation of Texinfo for further details.

Command line options

lilypond-book accepts the following command line options:

-f *format*
--format=*format*
: Specify the document type to process: html, latex, texi (the default) or docbook. If this option is missing, lilypond-book tries to detect the format automatically, see Sezione 3.5 [Filename extensions], pagina 26. Currently, texi is the same as texi-html.

-F *filter*
--filter=*filter*
: Pipe snippets through *filter*. lilypond-book will not –filter and –process at the same time. For example,

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

-h
--help
: Print a short help message.

-I *dir*
--include=*dir*
: Add *dir* to the include path. lilypond-book also looks for already compiled snippets in the include path, and does not write them back to the output directory, so in some cases it is necessary to invoke further processing commands such as makeinfo or latex with the same -I *dir* options.

-o *dir*
--output=*dir*
: Place generated files in directory *dir*. Running lilypond-book generates lots of small files that LilyPond will process. To avoid all that garbage in the source directory, use the '--output' command line option, and change to that directory before running latex or makeinfo.

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

...

`--skip-lily-check`
: Do not fail if no lilypond output is found. It is used for LilyPond Info documentation without images.

`--skip-png-check`
: Do not fail if no PNG images are found for EPS files. It is used for LilyPond Info documentation without images.

`--lily-output-dir=`*dir*
: Write lily-XXX files to directory *dir*, link into `--output` directory. Use this option to save building time for documents in different directories which share a lot of identical snippets.

`--info-images-dir=`*dir*
: Format Texinfo output so that Info will look for images of music in *dir*.

`--latex-program=`*prog*
: Run executable `prog` instead of `latex`. This is useful if your document is processed with `xelatex`, for example.

`--left-padding=`*amount*
: Pad EPS boxes by this much. *amount* is measured in millimeters, and is 3.0 by default. This option should be used if the lines of music stick out of the right margin.

  The width of a tightly clipped system can vary, due to notation elements that stick into the left margin, such as bar numbers and instrument names. This option will shorten each line and move each line to the right by the same amount.

`-P` *command*
`--process=`*command*
: Process LilyPond snippets using *command*. The default command is `lilypond`. `lilypond-book` will not `--filter` and `--process` at the same time.

`--pdf` Create PDF files for use with PDFLaTeX.

`--use-source-file-names`
: Write snippet output files with the same base name as their source file. This option works only for snippets included with `lilypondfile` and only if directories implied by `--output-dir` and `--lily-output-dir` options are different.

`-V`
`--verbose`
: Be verbose.

`-v`
`--version`
: Print version information.

Problemi noti e avvertimenti

The Texinfo command `@pagesizes` is not interpreted. Similarly, LaTeX commands that change margins and line widths after the preamble are ignored.

Only the first `\score` of a LilyPond block is processed.

3.5 Filename extensions

You can use any filename extension for the input file, but if you do not use the recommended extension for a particular format you may need to manually specify the output format; for details, see Sezione 3.4 [Invoking lilypond-book], pagina 24. Otherwise, `lilypond-book` automatically selects the output format based on the input filename's extension.

extension	**output format**
'`.html`'	HTML
'`.htmly`'	HTML
'`.itely`'	Texinfo
'`.latex`'	LaTeX
'`.lytex`'	LaTeX
'`.lyxml`'	DocBook
'`.tely`'	Texinfo
'`.tex`'	LaTeX
'`.texi`'	Texinfo
'`.texinfo`'	Texinfo
'`.xml`'	HTML

If you use the same filename extension for the input file than the extension `lilypond-book` uses for the output file, and if the input file is in the same directory as `lilypond-book` working directory, you must use `--output` option to make `lilypond-book` running, otherwise it will exit with an error message like "Output would overwrite input file".

3.6 lilypond-book templates

These templates are for use with `lilypond-book`. If you're not familiar with this program, please refer to Capitolo 3 [lilypond-book], pagina 14.

3.6.1 LaTeX

You can include LilyPond fragments in a LaTeX document.

```
\documentclass[]{article}

\begin{document}

Normal LaTeX text.

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

More LaTeX text, and options in square brackets.

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

3.6.2 Texinfo

You can include LilyPond fragments in Texinfo; in fact, this entire manual is written in Texinfo.

```
\input texinfo @node Top
@top

Texinfo text

@lilypond
```

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

More Texinfo text, and options in brackets.

@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>
Documents for lilypond-book may freely mix music and text.  For
example,
<lilypond>
\relative c'' {
  a4 b c d
}
</lilypond>
</p>

<p>
Another bit of lilypond, this time with options:

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

</body>
</html>
```

3.6.4 xelatex

```
\documentclass{article}
\usepackage{ifxetex}
\ifxetex
%xetex specific stuff
\usepackage{xunicode,fontspec,xltxtra}
\setmainfont[Numbers=OldStyle]{Times New Roman}
\setsansfont{Arial}
\else
```

```
%This can be empty if you are not going to use pdftex
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{mathptmx}%Times
\usepackage{helvet}%Helvetica
\fi
%Here you can insert all packages that pdftex also understands
\usepackage[ngerman,finnish,english]{babel}
\usepackage{graphicx}

\begin{document}
\title{A short document with LilyPond and xelatex}
\maketitle

Normal \textbf{font} commands inside the \emph{text} work,
because they \textsf{are supported by \LaTeX{} and XeteX.}
If you want to use specific commands like \verb+\XeTeX+, you
should include them again in a \verb+\ifxetex+ environment.
You can use this to print the \ifxetex \XeTeX{} command \else
XeTeX command \fi which is not known to normal \LaTeX .

In normal text you can easily use LilyPond commands, like this:

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

\noindent
and so on.

The fonts of snippets set with LilyPond will have to be set from
inside
of the snippet.  For this you should read the AU on how to use
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 Sharing the table of contents

These functions already exist in the OrchestralLily package:

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

For greater flexibility in text handling, some users prefer to export the table of contents from lilypond and read it into LaTeX.

Exporting the ToC from LilyPond

This assumes that your score has multiple movements in the same lilypond output file.

```
#(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))
}
```

Importing the ToC into LaTeX

In LaTeX, the header should include:

```
\usepackage{pdfpages}
\includescore{nameofthescore}
```

where `\includescore` is defined as:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \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 Alternative methods of mixing text and music

Other means of mixing text and music (without lilypond-book) are discussed in Sezione 4.4 [LilyPond output in other programs], pagina 37.

4 External programs

LilyPond can interact with other programs in various ways.

4.1 Point and click

Point and click lets you find notes in the input by clicking on them in the PDF viewer. This makes it easier to find input that causes some error in the sheet music.

When this functionality is active, LilyPond adds hyperlinks to the PDF file. These hyperlinks are sent to the web-browser, which opens a text-editor with the cursor in the right place.

To make this chain work, you should configure your PDF viewer to follow hyperlinks using the '`lilypond-invoke-editor`' script supplied with LilyPond.

For Xpdf on UNIX, the following should be present in '`xpdfrc`'[1]

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

The program '`lilypond-invoke-editor`' is a small helper program. It will invoke an editor for the special `textedit` URIs, and run a web browser for others. It tests the environment variable `EDITOR` for the following patterns,

`emacs` this will invoke

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

`gvim` this will invoke

```
gvim --remote +:line:normcolumn file
```

`nedit` this will invoke

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

The environment variable `LYEDITOR` is used to override this. It contains the command line to start the editor, where `%(file)s`, `%(column)s`, `%(line)s` is replaced with the file, column and line respectively. The setting

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

for `LYEDITOR` is equivalent to the standard emacsclient invocation.

The point and click links enlarge the output files significantly. For reducing the size of PDF and PS files, point and click may be switched off by issuing

```
\pointAndClickOff
```

in a '`.ly`' file. Point and click may be explicitly enabled with

```
\pointAndClickOn
```

Alternately, you may disable point and click with a command-line option:

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

Nota: You should always turn off point and click in any LilyPond files to be distributed to avoid including path information about your computer in the .pdf file, which can pose a security risk.

[1] On UNIX, this file is found either in '`/etc/xpdfrc`' or as '`.xpdfrc`' in your home directory.

4.2 Text editor support

There is support for different text editors for LilyPond.

Emacs mode

Emacs has a '`lilypond-mode`', which provides keyword autocompletion, indentation, LilyPond specific parenthesis matching and syntax coloring, handy compile short-cuts and reading LilyPond manuals using Info. If '`lilypond-mode`' is not installed on your platform, see below.

An Emacs mode for entering music and running LilyPond is contained in the source archive in the '`elisp`' directory. Do `make install` to install it to *elispdir*. The file '`lilypond-init.el`' should be placed to *load-path*'`/site-start.d/`' or appended to your '`~/.emacs`' or '`~/.emacs.el`'.

As a user, you may want add your source path (e.g. '`~/site-lisp/`') to your *load-path* by appending the following line (as modified) to your '`~/.emacs`'

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

Vim mode

For Vim, a filetype plugin, indent mode, and syntax-highlighting mode are available to use with LilyPond. To enable all of these features, create (or modify) your '`$HOME/.vimrc`' to contain these three lines, in order:

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

If LilyPond is not installed in the '`/usr/local/`' directory, change the path appropriately. This topic is discussed in Sezione "Other sources of information" in *Manuale di Apprendimento*.

Other editors

Other editors (both text and graphical) support LilyPond, but their special configuration files are not distributed with LilyPond. Consult their documentation for more information. Such editors are listed in Sezione "Easier editing" in *Informazioni generali*.

4.3 Converting from other formats

Music can be entered also by importing it from other formats. This chapter documents the tools included in the distribution to do so. There are other tools that produce LilyPond input, for example GUI sequencers and XML converters. Refer to the website for more details.

These are separate programs from `lilypond` itself, and are run on the command line; see ⟨undefined⟩ [Command-line usage], pagina ⟨undefined⟩ for more information. If you have MacOS 10.3 or 10.4 and you have trouble running some of these scripts, e.g. `convert-ly`, see Sezione "MacOS X" in *Informazioni generali*.

Problemi noti e avvertimenti

We unfortunately do not have the resources to maintain these programs; please consider them "as-is". Patches are appreciated, but bug reports will almost certainly not be resolved.

4.3.1 Invoking `midi2ly`

`midi2ly` translates a Type 1 MIDI file to a LilyPond source file.

MIDI (Music Instrument Digital Interface) is a standard for digital instruments: it specifies cabling, a serial protocol and a file format. The MIDI file format is a de facto standard format for exporting music from other programs, so this capability may come in useful when importing files from a program that has a converter for a direct format.

`midi2ly` converts tracks into Sezione "Staff" in *Guida al Funzionamento Interno* and channels into Sezione "Voice" in *Guida al Funzionamento Interno* contexts. Relative mode is used for pitches, durations are only written when necessary.

It is possible to record a MIDI file using a digital keyboard, and then convert it to '`.ly`'. However, human players are not rhythmically exact enough to make a MIDI to LY conversion trivial. When invoked with quantizing (`-s` and `-d` options) `midi2ly` tries to compensate for these timing errors, but is not very good at this. It is therefore not recommended to use `midi2ly` for human-generated midi files.

It is invoked from the command-line as follows,

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

Note that by 'command-line', we mean the command line of the operating system. See Sezione 4.3 [Converting from other formats], pagina 33, for more information about this.

The following options are supported by `midi2ly`.

`-a, --absolute-pitches`
: Print absolute pitches.

`-d, --duration-quant=`*`DUR`*
: Quantize note durations on *DUR*.

`-e, --explicit-durations`
: Print explicit durations.

`-h,--help`
: Show summary of usage.

`-k, --key=`*`acc`*`[:`*`minor`*`]`
: Set default key. *acc* > 0 sets number of sharps; *acc* < 0 sets number of flats. A minor key is indicated by `:1`.

`-o, --output=`*`file`*
: Write output to *file*.

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

`-t, --allow-tuplet=`*`DUR*`*`NUM/DEN`*
: Allow tuplet durations *DUR***NUM*/*DEN*.

`-v, --verbose`
: Be verbose.

`-V, --version`
: Print version number.

`-w, --warranty`
: Show warranty and copyright.

`-x, --text-lyrics`
: Treat every text as a lyric.

Problemi noti e avvertimenti

Overlapping notes in an arpeggio will not be correctly rendered. The first note will be read and the others will be ignored. Set them all to a single duration and add phrase markings or pedal indicators.

4.3.2 Invoking `musicxml2ly`

MusicXML is an XML dialect for representing music notation.

`musicxml2ly` extracts the notes, articulations, score structure, lyrics, etc. from part-wise MusicXML files, and writes them to a '`.ly`' file. It is invoked from the command-line.

It is invoked from the command-line as follows,

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

Note that by 'command-line', we mean the command line of the operating system. See Sezione 4.3 [Converting from other formats], pagina 33, for more information about this.

If the given filename is '`-`', `musicxml2ly` reads input from the command line.

The following options are supported by `musicxml2ly`:

`-a, --absolute`
: convert pitches in absolute mode.

`-h,--help`
: print usage and option summary.

`-l, --language=LANG`
: use LANG for pitch names, e.g. 'deutsch' for note names in German.

`--lxml` use the lxml.etree Python package for XML-parsing; uses less memory and cpu time.

`--nd --no-articulation-directions`
: do not convert directions (`^`, `_` or `-`) for articulations, dynamics, etc.

`--no-beaming`
: do not convert beaming information, use LilyPond's automatic beaming instead.

`-o,--output=`*`file`*
: set output filename to *file*. If *file* is '`-`', the output will be printed on stdout. If not given, *xml-file*'`.ly`' will be used.

`-r,--relative`
: convert pitches in relative mode (default).

`-v,--verbose`
: be verbose.

`--version`
: print version information.

`-z,--compressed`
: input file is a zip-compressed MusicXML file.

4.3.3 Invoking `abc2ly`

> **Nota:** This program is not supported, and may be remove from future versions of LilyPond.

ABC is a fairly simple ASCII based format. It is described at the ABC site:

`http://www.walshaw.plus.com/abc/learn.html`.

`abc2ly` translates from ABC to LilyPond. It is invoked as follows:

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

The following options are supported by `abc2ly`:

`-b,--beams=None`
: preserve ABC's notion of beams

`-h,--help`
: this help

`-o,--output=`*file*
: set output filename to *file*.

`-s,--strict`
: be strict about success

`--version`
: print version information.

There is a rudimentary facility for adding LilyPond code to the ABC source file. If you say:

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

This will cause the text following the keyword 'voices' to be inserted into the current voice of the LilyPond output file.

Similarly,

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

will cause the text following the 'slyrics' keyword to be inserted into the current line of lyrics.

Problemi noti e avvertimenti

The ABC standard is not very 'standard'. For extended features (e.g., polyphonic music) different conventions exist.

Multiple tunes in one file cannot be converted.

ABC synchronizes words and notes at the beginning of a line; `abc2ly` does not.

`abc2ly` ignores the ABC beaming.

4.3.4 Invoking `etf2ly`

> **Nota:** This program is not supported, and may be remove from future versions of LilyPond.

ETF (Enigma Transport Format) is a format used by Coda Music Technology's Finale product. `etf2ly` will convert part of an ETF file to a ready-to-use LilyPond file.

It is invoked from the command-line as follows.

```
etf2ly [option]... etf-file
```

Note that by 'command-line', we mean the command line of the operating system. See Sezione 4.3 [Converting from other formats], pagina 33, for more information about this.

The following options are supported by `etf2ly`:

`-h,--help`
: this help

`-o,--output=`*FILE*
: set output filename to *FILE*

`--version`
: version information

Problemi noti e avvertimenti

The list of articulation scripts is incomplete. Empty measures confuse `etf2ly`. Sequences of grace notes are ended improperly.

4.3.5 Other formats

LilyPond itself does not come with support for any other formats, but some external tools can also generate LilyPond files. These are listed in Sezione “Easier editing” in *Informazioni generali*.

4.4 LilyPond output in other programs

This section shows methods to integrate text and music, different than the automated method with `lilypond-book`.

Many quotes from a large score

If you need to quote many fragments from a large score, you can also use the clip systems feature, see Sezione “Extracting fragments of music” in *Guida alla Notazione*.

Inserting LilyPond output into OpenOffice.org

LilyPond notation can be added to OpenOffice.org with OOoLilyPond.

Inserting LilyPond output into other programs

To insert LilyPond output in other programs, use `lilypond` instead of `lilypond-book`. Each example must be created individually and added to the document; consult the documentation for that program. Most programs will be able to insert LilyPond output in ‘`PNG`’, ‘`EPS`’, or ‘`PDF`’ formats.

To reduce the white space around your LilyPond score, use the following options

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

{ c1 }
```

To produce a useful ‘`EPS`’ file, use

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

‘PNG’:
lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts --png myfile.ly
```

4.5 Independent `includes`

Some people have written large (and useful!) code that can be shared between projects. This code might eventually make its way into LilyPond itself, but until that happens, you must download and `\include` them manually.

4.5.1 MIDI articulation

LilyPond can be used to produce MIDI output, for “proof-hearing” what has been written. However, only dynamics, explicit tempo markings, and the notes and durations themselves are produced in the output.

The *articulate* project is one attempt to get more of the information in the score into he MIDI. It works by shortening notes not under slurs, to ‘articulate’ the notes. The amount of shortening depends on any articulation markings attached to a note: staccato halves the note

value, tenuto gives a note its full duration, and so on. The script also realises trills and turns, and could be extended to expand other ornaments such as mordents.

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

Problemi noti e avvertimenti

Its main limitation is that it can only affect things it knows about: anything that is merely textual markup (instead of a note property) is still ignored.

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 brani 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 altre questioni da tenere a mente quando si scrivono file di input di LilyPond.

- Se fai un errore? La struttura di un file LilyPond può rendere l'individuazione di certi errori più facile (o più difficile).
- Se vuoi inviare i tuoi file di input a qualcuno? E se vuoi modificare i tuoi file di input dopo qualche anno? Alcuni file di input di LilyPond sono comprensibili a prima vista; altri ti possono lasciare perplesso per un'ora.
- Se vuoi aggiornare il tuo file per poterlo usare con una versione più recente di LilyPond? La sintassi di input cambia di tanto in tanto mentre LilyPond si evolve. 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 per poter essere aggiornati in modo più semplice (o più difficile).

5.1 Consigli generali

Ecco alcuni consigli che possono aiutarti ad 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. Per esperienza personale, è piuttosto 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 "Bar and bar number checks" in *Guida alla Notazione*, Sezione "Octave checks" in *Guida alla Notazione*. Se includi i controlli ogni tanto, allora 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, magari solo una o due volte. Se la musica è molto complessa, ad 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 conviene affatto 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 non sono presenti dei commenti.
- **Indenta le parentesi graffe**. Molti problemi sono causati da uno squilibrio nel numero di `{` e `}`.
- **Esplicita le durate** all'inizio delle sezioni e delle variabili. Se specifichi `c4 d e` all'inizio di un fraseggio (innvece di `c d e` soltanto) puoi evitare alcuni problemi quando riarrangi la musica successivamente.
- **Separa le modifiche manuali (tweak)** dalle definizioni musicali. Vedi Sezione "Ridurre l'input grazie a variabili e funzioni" in *Manuale di Apprendimento*, e Sezione "Style sheets" in *Manuale di Apprendimento*.

5.2 Scrivere musica esistente

Se stai scrivendo della musica da una partitura esistente (ovvero il brano di 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 quando ne completi uno. 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 uno strumento traspositore in una variabile, si consiglia di avvolgere le note tra parentesi

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

  (dove `altezza-naturale` è l'altezza dello strumento) in modo che la musica contenuta nella variabile sia effettivamente in Do. Puoi trasporla all'indietro quando la variabile viene usata, se richiesto, ma potresti non desiderarlo (ad esempio quando si stampa una partitura in intonazione reale, quando si converte 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 verso/dal Do. Questo significa che le uniche altre tonalità che userai sono 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 che più probabilmente cambierà; la definizione di `violin` è molto improbabile che 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, ma 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 devi fare qualche ricerca per individuare l'origine del problema.

Gli strumenti più potenti per questo scopo sono il commento della linea singola (indicato da `%`) e il commento di blocco (indicato da `%{ ... %}`). Se non sai dove sta 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 hai qualcosa che funziona.

Nel caso più estremo, potresti finire con soltanto

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

(in altre parole, un file senza musica)

Se accade questo, 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 piano piano a decommentare 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 Makefiles

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 file e quali comandi bisogna 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.

Ci sono situazioni in cui è una buona idea creare un `Makefile` per il proprio progetto, per proprio comodo o come cortesia per altri che potrebbero 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`). Makefiles 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 quel che può fare.

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

Il primo esempio è per un'opera per orchestra in quattro movimenti e ha la seguente struttura di directory:

```
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 ../definitions.ily
\include ../Notes/cello.ily
```

Il makefile avrà i target di `score` (l'intero brano in partitura completa), `movements` (movimenti individuali in partitura completa), e `parts` (parti individuali 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 complicazioni specifiche nella 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 14

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 in LaTeX documents 18

A

ABC 35
aggiornare i vecchi file di input 11
Aggiornare un file di LilyPond 11
aiuto, linea di comando 2
avvertimento 7

C

cartella, dirigere l'output in 4
Coda Technology 36
coloring, syntax 33
convert-ly 11

D

docbook 14
DocBook, music in 14
documents, adding music to 14
dvips 24

E

`Easier editing` 33, 37
editors 33
emacs 33
enigma 36
EPS (Encapsulated PostScript) 3
errore 7
Errore di programmazione 7
errore fatale 7
errore Scheme 7
errori, formato del messaggio 7
`Esempi minimi` 41
ETF 36
External programs, generating LilyPond files 37

F

file size, output 32
Finale 36
formato di output, impostare il 3

H

html 14
HTML, music in 14

I

Invocare `lilypond` 2
invoking dvips 24

L

LANG 5
latex 14
LaTeX, music in 14
LILYPOND_DATADIR 5
linea di comando, opzioni di 2

M

`MacOS X` 1, 14, 33
make 41
makefiles 41
`Manuals` 1
messaggi di errore 7
MIDI 33
modes, editor 33
musicology 14
MusicXML 35

N

nome del file di output, impostare 4

O

OpenOffice.org 37
opzioni della linea di comando per `lilypond` 2
outline fonts 24

P

paper-size, linea di comando 2
percorso di ricerca 4
point and click 32
Portable Document Format (PDF) 4
Portable Network Graphics (PNG) 4
PostScript 4
Postscript, incapsulato 3
PostScript, output 3
preview image 20
preview, linea di comando 3
punta e clicca, linea di comando 2

R

ricerca dei file 4

S

safe, linea di comando 2
Scheme, estrazione di 3
Sospeso (core dumped) 7
SVG (Scalable Vector Graphics) 3
switches 2
syntax coloring 33

T

texi 14
texinfo 14
Texinfo, music in 14
thumbnail 20
titling and lilypond-book 18
titling in HTML 20
traccia di chiamata 7
traccia, Scheme 7
type1 fonts 24

V

vim 33