LilyPond

El gravador de música

Utilització

L'equip de desenvolupadors del LilyPond

Aquest fitxer explica com executar els programes que es distribueixen amb el LilyPond versió 2.21.4. A més a més suggereix certes "bones pràctiques" per a una utilització eficient.

Per a més informació sobre la forma en la qual aquest manual es relaciona amb la resta de la documentació, o per llegir aquest manual en altres formats, consulteu Secció "Manuals" in *Informació general*.

Si us falta algun manual, trobareu tota la documentació a `http://lilypond.org/`.

Copyright © 1999–2020 pels autors.*La traducció de la següent nota de copyright s'ofereix com a cortesia per a les persones de parla no anglesa, però únicament la nota en anglès té validesa legal.*

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

> S'atorga permís per copiar, distribuir i/o modificar aquest document sota els termes de la Llicència de Documentació Lliure de GNU, versió 1.1 o qualsevol posterior publicada per la Free Software Foundation; sense cap de les seccions invariants. S'inclou una còpia d'aquesta llicència dins de la secció titulada "Llicència de Documentació Lliure de 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 a la versió del LilyPond 2.21.4

Índex General

1 Execució del LilyPond 1

1.1 Utilització normal 1
1.2 Utilització des de la línia d'ordres 1
Invocació del `lilypond` 1
Instruccions estàndard de la línia d'ordres 1
Opcions bàsiques de la línia d'ordres per al LilyPond 2
Opcions avançades de la línia d'ordres per al LilyPond 4
Variables d'entorn 9
El LilyPond a una gàbia de chroot 9
1.3 Missatges d'error 11
1.4 Errors comuns 12
La música se surt de la pàgina 12
Apareix un pentagrama de més 12
Missatge d'error Unbound variable % 13
Missatge d'error FT_Get_Glyph_Name 13
Advertiment sobre que les afinitats del pentagrama sols han de decrèixer 14
Missatge d'error Unexpected new `\new` 14

2 Actualització de fitxers amb `convert-ly` 16

2.1 Perquè canvia la sintaxi? 16
2.2 Invocació de `convert-ly` 16
2.3 Opcions de la línia d'ordres per a `convert-ly` 17
2.4 Problemes amb `convert-ly` 18
2.5 Conversions manuals 18

3 Running `lilypond-book` 20

3.1 An example of a musicological document 20
3.2 Integrating music and text 24
3.2.1 LaTeX 24
3.2.2 Texinfo 26
3.2.3 HTML 27
3.2.4 DocBook 27
3.3 Music fragment options 28
3.4 Invoking `lilypond-book` 31
3.5 Filename extensions 34
3.6 lilypond-book templates 34
3.6.1 LaTeX 34
3.6.2 Texinfo 35
3.6.3 html 35
3.6.4 xelatex 36
3.7 Sharing the table of contents 37
3.8 Alternative methods of mixing text and music 38

4 External programs 39

4.1 Point and click 39
4.1.1 Configuring the system 39
Using Xpdf 39

Using GNOME 2 40
Using GNOME 3 40
Extra configuration for Evince 40
Enabling point and click 40
Selective point-and-click 41
4.2 Text editor support 41
Emacs mode 42
Vim mode 42
Other editors 42
4.3 Converting from other formats 42
4.3.1 Invoking `midi2ly` 42
4.3.2 Invoking `musicxml2ly` 44
4.3.3 Invoking `abc2ly` 45
4.3.4 Invoking `etf2ly` 46
4.3.5 Other formats 46
4.4 LilyPond output in other programs 46
4.4.1 LuaTex 46
4.4.2 OpenOffice and LibreOffice 46
4.4.3 Other programs 47
4.5 Independent `includes` 47
4.5.1 MIDI articulation 47

5 Suggeriments per escriure fitxers d'entrada 48
5.1 Suggeriments de tipus general 48
5.2 Gravació de música existent 49
5.3 Projectes grans 49
5.4 Solució de problemes 50
5.5 Make i els Makefiles 50

Annex A GNU Free Documentation License 57

Annex B Índex del LilyPond 64

1 Execució del LilyPond

Aquest capítol detalla els aspectes tècnics de l'execució del LilyPond.

1.1 Utilització normal

Gairebé tots els usuaris executen el LilyPond per mitjà d'una interfície gràfica; consulteu Secció "Tutorial" in *Manual d'aprenentatge* si encara no l'heu llegit.

1.2 Utilització des de la línia d'ordres

Aquesta secció conté informació addicional sobre l'ús del LilyPond a la línia d'ordres. Aquesta forma pot ser preferible per passar-li al programa algunes opcions addicionals. A més a més, existeixen alguns programes complementaris 'de suport' (com ara `midi2ly`) que sols estan disponibles a la línia d'ordres.

En parlar de la 'línia d'ordres', ens referim a la consola del sistema operatiu. Els usuaris del Windows possiblement estiguin més familiaritzats amb els termes 'finestra del MS-DOS' o 'línia de comandes'; Els usuaris del MacOS X potser que estiguin més familiaritzats amb els termes 'terminal' o 'consola'. Aquests podrien requerir algunes configuracions addicionals i haurien de consultar també l'apartat Secció "MacOS X" in *Informació general*.

La descripció de l'ús d'aquesta part dels sistemes operatius excedeix l'àmbit d'aquest manual; us preguem que consulteu altres documents sobre aquest tema si no us resulta familiar la línia d'ordres.

Invocació del `lilypond`

L'executable `lilypond` es pot cridar des d'una línia d'ordres de la manera següent:

```
lilypond [opció]... fitxer...
```

Quan s'invoca amb un nom de fitxer sense extensió, es prova en primer lloc amb la extensió `.ly`. Per llegir l'entrada des de stdin, utilitzeu un guió (`-`) en substitució de *fitxer*.

Quan es processa `archivo.ly`, la sortida resultant són els fitxers `fitxer.ps` i `fitxer.pdf`. Es poden especificar diversos fitxers; cadascú d'ells es processarà de forma independent[1].

Si `fitxer.ly` conté més d'un bloc `\score`, la resta de les partitures s'obtindran com a sortida en fitxers numerats, començant per `fitxer-1.pdf`. A més, el valor de `output-suffix` (sufix de sortida) s'inserirà entre el nom base i el número. Un fitxer de sortida que contingui

```
#(define output-suffix "violí")
\score { ... }
#(define output-suffix "violoncel")
\score { ... }
```

produirà com a sortida *base*`-violí.pdf` i *base*`-violoncel-1.pdf`.

Instruccions estàndard de la línia d'ordres

Si la vostra terminal (o finestra d'ordres) contempla les redireccions normals, potser us siguin d'utilitat les següents instruccions per redirigir la sortida de la consola d'un fitxer:

- `lilypond fitxer.ly 1>sortidaestandard.log` per redirigir la sortida normal
- `lilypond fitxer.ly 2>sortidaderror.log` per redirigir els missatges d'error
- `lilypond fitxer.ly &>tot.log` per redirigir tota la sortida

[1] L'estat del GUILE no es restableix després de processar un fitxer `.ly`, per la qual cosa heu de tenir cura de no modificar cap valor predeterminat des de dins del Scheme.

Consulteu la documentació del vostre intèrpret d'ordres per veure si contempla aquestes opcions, o si la sintaxi és diferent. Observeu que són instruccions de l'intèrpret d'ordres i que no tenen res a veure amb el LilyPond.

Opcions bàsiques de la línia d'ordres per al LilyPond

Estan contemplades les opcions següents:

`-d, --define-default=`*`variable=valor`*
: Vegeu [Opcions avançades de la línia d'ordres per al LilyPond], pàgina 4.

`-e, --evaluate=`*`expressió`*
: Avalua l'*expressió* del Scheme abans d'analitzar els fitxers `.ly`. Es poden passar diverses opcions `-e`, que s'avaluaran en seqüència.

  L'expressió s'avaluarà al mòdul `guile-user`, de manera que si voleu usar definicions dins d'*expressió*, heu d'utilitzar

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

  a la línia d'ordres, i incloure

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

  al principi del fitxer `.ly`.

  > **Nota:** Els usuaris de Windows han d'utilitzar cometes dobles en comptes de cometes simples.

`-f, --format=`*`format`*
: quins formats s'han d'escriure. Les opcions per a `format` són `ps`, `pdf`, i `png`.

  Exemple: `lilypond -fpng` *`fitxer`*`.ly`

`-h, --help`
: Mostra un resum de les formes de utilització.

`-H, --header=`*`CAMP`*
: Bolca un camp de capçalera al fitxer `NOMBASE.`*`CAMP`*

`-i, --init=`*`archivo`*
: Establir el fitxer d'inici a *fitxer* (predeterminat: `init.ly`).

`-I, --include=`*`directori`*
: Afegir el *directori* a la ruta de cerca de fitxers d'entrada.

  Es poden escriure diverses opcions -I. La cerca s'inicia al primer directori definit, i si el fitxer que s'ha d'incloure no es troba, la cerca continua als directoris següents.

`-j, --jail=`*`usuari,grup,gàbia,directori`*
: Executar `lilypond` a una gàbia de chroot.

  L'opció `--jail` (gàbia) proporciona una alternativa més flexible a l'opció `-dsafe` quan el procés de tipografia del LilyPond està disponible a un servidor web o quan el LilyPond executa instruccions enviades per fonts externes (vegeu [Opcions avançades de la línia d'ordres per al LilyPond], pàgina 4).

  L'opció `--jail` funciona canviant l'arrel de `lilypond` a *gàbia* just o abans de començar el procés de compilació en sí. Si es fa això es canvien l'usuari i el grup als que s'han donat a l'opció, i el directori actual es canvia a *directori*. Aquesta instal·lació garanteix que no és possible, al menys en teoria, escapar a la gàbia. Observeu que perquè funcioni `--jail`, s'ha d'executar `lilypond` com root, cosa que normalment es pot fer d'una forma segura utilitzant `sudo`.

La instal·lació d'una gàbia pot ser un assumpte relativament complex, atès que hem d'assegurar-nos que el LilyPond pot trobar *dins* de la pròpia gàbia tot el que necessita per poder compilar la font. Una típica configuració de gàbia de chroot consta dels següents elements:

Preparació d'un sistema de fitxers separat
: S'ha de crear un sistema de fitxers separat per al LilyPond, de forma que es pugui muntar amb opcions segures com `noexec`, `nodev` i `nosuid`. D'aquesta forma, és impossible executar programes o escriure directament a un dispositiu des del LilyPond. Si no voleu crear una partició separada, tan sols té que crear un fitxer d'una mida raonable i usar-lo per muntar un dispositiu loop. El sistema de fitxers separat garanteix també que el LilyPond mai no pugui escriure en un espai major del què se li permeti.

Preparar un usuari separat
: Es pot usar un usuari i grup separats (diguem-ne `lily`/`lily`) amb pocs privilegis per executar el LilyPond dins d'una gàbia. Hauria d'existir un sols directori amb permisos d'escriptura per a aquest usuari, i s'ha de passar el valor *directori*.

Preparació de la gàbia
: El LilyPond necessita llegir alguns fitxers mentre s'executa, Tots aquests fitxers s'han de copiar dins de la gàbia, sota la mateixa ruta en la qual apareixen al sistema de fitxers real de root. Tot el contingut de la instal·lació del LilyPond (per exemple `/usr/share/lilypond`) s'ha de copiar.

  Si sorgeixen problemes, la forma més senzilla de rastrejar-los és executar el LilyPond usant `strace`, cosa que li permetrà determinar quins fitxers falten.

Execució del LilyPond
: Dins d'una gàbia muntada amb `noexec` és impossible executar cap programa extern. Per tant, el LilyPond s'ha d'executar amb un backend que no necessiti un programa extern. Com ja hem mencionat, s'ha d'executar amb privilegis del superusuari (que per suposat perdrà immediatament), possiblement usant `sudo`. També de CPU que el LilyPond pot usar (per exemple usant `ulimit -t`), i, si el vostre sistema operatiu ho contempla, la mida de la memòria que es pot reservar. Vegeu també [El LilyPond a una gàbia de chroot], pàgina 9.

`-l, --loglevel=`*NIVELL*
: Fixa el grau en el qual la sortida de consola és neta al nivell *NIVELL*. Els valors possibles són:

  `NONE` Cap sortida en absolut, ni tan sols missatges d'error.

  `ERROR` Sols missatges d'error, cap advertiment o indicacions de progrés.

  `WARN` Advertiments i missatges d'error, no de progrés.

  `BASIC_PROGRESS`
  : Missatges de progrés bàsics (èxit), advertiment i errors.

  `PROGRESS` Tots els missatges de progrés, advertiments i errors.

  `INFO (predeterminat)`
  : Missatges de progrés, advertiments, errors i informació d'execució addicional.

`DEBUG` Tots els missatges possibles, fins i tot la informació detallada de depuració.

`-o, --output=`*`FITXER`* o *`CARPETA`*
Estableix el nom del fitxer de sortida predeterminat a *FITXER* o, si hi ha una carpeta amb aquest nom, dirigeix la sortida cap a *CARPETA*, agafant el nom de fitxer del document d'entrada. S'afegeix el sufix corresponent (per exemple, `.pdf` per a PDF) als dos casos.

`--ps` Generar PostScript.

`--png` Genera imatges de les pàgines en format PNG. Això implica `--ps`. La resolució en PPP de la imatge es pot establir amb

```
-dresolution=110
```

`--pdf` Genera PDF. Implica `--ps`.

`-v, --version`
Mostra la informació de la versió.

`-V, --verbose`
Sigues detallat: mostra les rutes completes de tots els fitxers que se llegeixen, i dóna informació cronomètrica.

`-w, --warranty`
Mostra la garantia del GNU LilyPond (no ve amb **CAP GARANTIA**!).

Opcions avançades de la línia d'ordres per al LilyPond

`-d`*`[nom-de-opció]=[valor]`*`, --define-default=`*`[nom-de-opció]=[valor]`*
Estableix la funció del Scheme interna equivalent a *valor*.

```
-dbackend=svg
```

Si no es proporciona cap *valor*, s'usa el valor predeterminat Per desactivar una opció es pot anteposar `no-` a la *variable*, per exemple:

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

és el mateix que

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

Estan contemplades les següents opcions junt als seus respectius valors predeterminats:

Símbol	**Valor**	**Explicació/Opcions**
`anti-alias-factor` (factor d'antiàlies)	`1`	Renderitza a una major resolució (utilitzant el factor donat) i redueix l'escala del resultat per així evitar 'escales' a les imatges `PNG`.
`aux-files` (fitxers auxiliars)	`#t`	Crea fitxeres `.tex`, `.texi`, `.count` al 'backend' `EPS`.
`backend`	`ps`	Selecciona un 'rerefons'. Els fitxers (l'opció predeterminada) inclouen els tipus tipogràfics de lletra `TTF`, `Type1` i `OTF`. No es fa cap subconjunt d'aquests tipus de lletra. L'ús de conjunts de caràcters 'orientals' pot produir fitxers molts grans.

	`eps`	PostScript encapsulat. Bolca cada pàgina o sistema com un fitxer `EPS` diferent, sense tipus tipogràfics de lletra, i com un fitxer `EPS` enquadernat amb totes les pàgines o sistemes que inclouen els tipus de lletra. Utilitzat com a opció predeterminada per part de `lilypond-book`.
	`null`	No produeixes cap partitura impresa a la sortida; té el mateix efecte que `-dno-print-pages`.
	`svg`	Gràfics vectorials escalables. Crea un únic fitxer `SVG`, sense tipus tipogràfics de lletra incrustats, per a cada pàgina de sortida. Es recomana instal·lar el tipus de lletra Century Schoolbook, que està inclòs a la instal·lació del LilyPond, per a un renderitzat òptim. Sota l'UNIX, bastarà amb que copieu aquests fitxers de tipus de lletra del directori del Lilypond (normalment `/usr/share/lilypond/VERSION/fonts/otf/`) al directori `~/.fonts/`. La sortida `SVG` hauria de ser compatible amb qualsevol editor o client de SVG. També hi ha una opció `svg-woff` (vegeu més avall) per usar els fitxers de tipus de lletra woff al 'rerefons' SVG.
	`scm`	Bolcat de les instruccions de dibuix internes basades en Scheme, en brut.
`check-internal-types` n	`#f`	Comprova el tipus de cada assignació de propietats.
`clip-systems` (retalla els sistemes de pentagrames)	`#f`	Genera fragments d'imatge retallats d'una partitura.
`datadir` (directori de dades)		Prefix dels fitxers de dades (sols lectura).
`debug-gc`	`#f`	Bolca estadístiques de depuració de memòria.
`debug-gc-assert-parsed-dead`	`#f`	Per a la depuració de memòria: assegura't que totes les referències a objectes analitzats estiguin mortes. És una opció interna, i s'activa automàticament per a `` `-ddebug-gc' ``.
`debug-lexer`	`#f`	Depuració de l'analitzador lèxic flex.
`debug-page-breaking-scoring`	`#f`	Bolca les partitures per a moltes configuracions de salts de pàgina diferents.

`debug-parser`	`#f` f	Depuració de l'analitzador sintàctic bison.
`debug-property-callbacks`	`#f`	Depuració de les cadenes cícliues de funcions de callback.
`debug-skylines`	`#f`	Depuració de les línies de horitzó.
`delete-intermediate-files`	`#t`	Elimina els fitxers intermedis `.ps` inútils que es creen durant la compilació.
`dump-cpu-profile`	`#f`	Bolcar informació de comptabilització del temps (dependent del sistema).
`dump-profile`	`#f`	Bolca la informació de memòria i de temps de cada fitxer.
`dump-signatures`	`#f`	Bolca les signatures de sortida de cada sistema. Usat per a les proves de regressió.
`eps-box-padding`	`#f`	Omple la vora esquerra de la capsa contenidora de l'EPS de sortida en la quantitat donada (en mm).
`gs-load-fonts`	`#f`	Carrega els tipus tipogràfics de lletra a través del Ghostscript.
`gs-load-lily-fonts`	`#f`	Carrega sols els tipus de lletra del LilyPond per mitjà del Ghostscript.
`gui`	`#f`	S'executa silenciosament i es redirigeix tota la sortida a un fitxer de registre.

Nota per als usuaris del Windows: De manera predeterminada, `lilypond.exe` dirigeix tota la sortida de la informació d'avenç cap a la finestra de consola, `lilypond-windows.exe` no ho fa i retorna un indicador del sistema, sense cap indicació d'avenç, immediatament en la línia d'ordres. L'opció `-dgui` es pot usar en aquest cas per redirigir la sortida a un fitxer de registre.

`help`	`#f`	Mostra aquesta ajuda
`include-book-title-preview`	`#t`	Inclou els títols de llibre a les imatges de vista prèvia.
`include-eps-fonts`	`#t`	Incloure els tipus tipogràfics de fonts als fitxers EPS de cadascú dels sistemes.
`include-settings`	`#f`	Inclou el fitxer dels ajustos globals, s'inclou abans que la partitura es processi.
`job-count`	`#f`	Processa en paral·lel, usant el nombre de tasques donat.

`log-file`	`#f` `[fitxer]`	Si es dóna a una cadena `fitxer` como a segon argument, redirigeix la sortida al fitxer de registre `fitxer.log`.
`max-markup-depth`	`1024`	Profunditat màxima de l'arbre de l'etiquetatge. Si un etiquetatge té més nivells, suposa que no acabarà per sí mateix, imprimint un advertiment i retornant en el seu lloc un element d'etiquetatge nul.
`midi-extension`	`"midi"`	Fixa l'extensió de fitxer predeterminat per al fitxer de sortida MIDI a la cadena donada.
`music-strings-to-paths`	`#f`	Converteix les cadenes de text a rutes quan els glifs pertanyen a un tipus de lletra de tipografia musical.
`paper-size`	`\"a4\"`	Estableix la mida predeterminada del paper. Observeu que la cadena ha d'anar tancada entre cometes dobles.
`pixmap-format`	`png16m`	Fixa el format de sortida del Ghostsript per a les imatges de píxels.
`point-and-click`	`#f`	Afegeix enllaços d''apuntar i clicar' a la sortida `PDF`. Vegeu Secció 4.1 [Point and click], pàgina 39.
`preview`	`#f`	Crea imatges de vista prèvia a més de la sortida normal.

Aquesta opció està contemplada per tots els 'rerefons': `pdf`, `png`, `ps`, `eps` i `svg`, però no per `scm`. Genera un fitxer de sortida, en la forma `elmeuFitxer.preview.extensió`, que conté els títols i el primer sistema de la música. Si s'estan utilitzant blocs `\book` o `\bookpart`, apareixen a la sortida els títols de `\book`, `\bookpart` o `\score`, inclòs el primer sistema de cada bloc `\score` si la variable de `\paper` `print-all-headers` està fixada al valor `#t`.

Per suprimir la sortida actual, utilitzeu les opcions `-dprint-pages` o `-dno-print-pages` segons les vostres necessitats.

`print-pages`	`#t`	Genera pàgines completes (és l'opció predeterminada). És útil `-dno-print-pages` en combinació amb `-dpreview`.
`profile-property-accesses`	`#f`	Conserva les estadístiques de les crides de funció `get_property()`.
`protected-scheme-parsing`	`#t`	Continua quan es capten a l'analitzador sintàctic errors del Scheme encastat. Si es fixa a `#f`, detenir-se quan hi hagi errors i imprimir un registre de traça de pila.

`read-file-list`	`#f` `[fitxer]`	Especifica el nom d'un fitxer que conté una llista de fitxers d'entrada per processar.
`relative-includes`	`#f`	Quan es processa una instrucció `\include`, cerca el fitxer inclòs de forma relativa al fitxer actual (enlloc del fitxer principal).
`resolution`	`101`	Fixa la resolució per generar imatges de píxels `PNG` al valor donat (en ppp).
`safe`	`#f`	No confiïs en l'entrada `.ly`.

Quan el servei de tipografia està disponible a través d'un servidor web, **S'HAN DE** passar les opcions `--safe` o `--jail`. L'opció `--safe` evita que el codi del Scheme faci un desastre, per exemple:

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

L'opció `-dsafe` funciona avaluant les expressions del Scheme en línia dins d'un mòdul segur especial. Deriva del mòdul `safe-r5rs` del GUILE, però a més afegeix unes quantes funcions de l'API del LilyPond que estan relacionades en `scm/safe-lily.scm`.

A més, el mode segur prohibeix les directives `\include` i desactiva la utilització de barres invertides a les cadene de TEX. A més, no és possible importar variables del LilyPond dins del Scheme quan s'està em mode segur.

`-dsafe` *no* detecta la sobreutilització de recursos, per la qual cosa encara és possible fer que el programa es pengi indefinidament, per exemple subministrant estructures de dades cícliques en el rerefons. Per això, si esteu usant el LilyPond en un servidor web accessible públicament, el procés s'ha de limitar tant en l'ús de memòria com de CPU.

El mode segur evita que es puguin compilar molts fragments de codi útils.

L'opció `--jail` és una alternativa encara més segura, però requereix més feina per a la seva configuració. Vegeu [Opcions bàsiques de la línia d'ordres per al LilyPond], pàgina 2.

`separate-log-files`	`#f`	Per als fitxers d'entrada `FITXER1.ly`, `FITXER2.ly`, etc., treu les dades de registre cap als fitxers `FITXER1.log`, `FITXER2.log`...
`show-available-fonts`	`#f`	Llista tots els noms dels tipus tipogràfics de lletra disponibles.
`strict-infinity-checking`	`#f`	Força una terminació abrupta si es troben les excepcions de punt flotant `Inf` i `NaN`.
`strip-output-dir`	`#t`	No usis els directoris dels fitxers d'entrada en construir els noms dels fitxers de sortida.

`strokeadjust`	`#f`	Força l'ajust dels traços de PostScript. Aquesta opció és rellevant principalment quan es genera un `PDF` a partir de la sortida de PostScript (l'ajust del traç està en general activat automàticament per a dispositius de mapa de punts de baixa resolució). Sense aquesta opció, els visors de `PDF` tendeixen a produir amplades de plica molt poc consistents a les resolucions típiques de les pantalles d'ordinador. L'opció no afecta de forma molt significativa a la qualitat de la impressió i causa grans increments a la mida del fitxer `PDF`.
`svg-woff`	`#f`	Usar fitxers de tipus tipogràfic de lletra de woff al rerefons SVG.
`verbose`	`#f`	Sortida detallada, és a dir el nivell de registre en DEBUT (sols lectura).
`warning-as-error`	`#f`	Canvia tots els missatges d'advertiment i de 'error de programació' a errors.

Variables d'entorn

`lilypond` reconeix les següents variables d'entorn:

`LILYPOND_DATADIR`
: Especifica un directori en el qual els missatges de localització i de dades es buscaran de forma predeterminada. El directori ha de contenir subdirectoris anomenats `ly/`, `ps/`, `tex/`, etc.

`LANG`
: Selecciona l'idioma dels missatges d'advertiment.

`LILYPOND_LOGLEVEL`
: Nivell de registre predeterminat. Si el LilyPond es crida sense cap nivell de registre explícit (és a dir, sense opció de línia d'ordres `--loglevel`), s'usa aquest valor.

`LILYPOND_GC_YIELD`
: Una variable, com a percentatge, que ajusta el comportament de l'administració de memòria. Amb valors més alts, el programa usa més memòria; amb valors més baixos, usa més temps de CPU. El valor predeterminat és `70`.

El LilyPond a una gàbia de chroot

La preparació del servidor perquè executi el LilyPond a una gàbia de chroot és una tasca molt complicada. Els passos estan relacionats més avall. Els exemples que apareixen en cadascú dels passos son vàlids per a Ubuntu GNU/Linux, i poden requerir l'ús de `sudo` segons correspongui.

- Instal·leu els paquets necessaris: el LilyPond, el Ghostscript i l'ImageMagick.
- Creeu un usuari nou amb el nom de `lily`:

  ```
  adduser lily
  ```

  Això també crearà un nou grup per a l'usuari `lily`, i una carpeta personal, `/home/lily`
- A la carpeta personal de l'usuari `lily`, creeu un fitxer per usar-lo com a sistema de fitxers separat:

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

  Aquest exemple crea un fitxer de 200MB per al seu ús com el sistema de fitxers de la gàbia.

- Creeu un dispositiu loop, feu un sistema de fitxers i munteu-lo, després creeu una carpeta que es pugui escriure per l'usuari `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
```

- En la configuració dels servidors, JAIL serà `/mnt/lilyloop` i DIR serà `/lilyhome`.
- Creeu un gran arbre de directoris dins de la gàbia copiant els fitxers necessaris, com es mostra en el guió d'exemple que apareix més avall.

  Podeu usar `sed` per crear els fitxeres de còpia necessaris per a un executable donat:

```
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
```

Guió d'exemple per a l'Ubuntu 8.04 de 32 bits

```
#!/bin/sh
## aquí es fixen els valors predeterminats

username=lily
home=/home
loopdevice=/dev/loop0
jaildir=/mnt/lilyloop
# prefix (sense la barra inicial!)
lilyprefix=usr/local
# el directori en el qual el LilyPond es troba instal·lat en el sistema
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

# Ara la màgia de copiar les biblioteques
```

```
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

# Els fitxers compartits per al ghostcript...
      cp -L -r /usr/share/ghostscript usr/share
# Els fitxers compartits per a l'ImageMagick
      cp -L -r /usr/lib/ImageMagick* usr/lib

### Ara, suposant que tenim test.ly a /mnt/lilyloop/lilyhome,
### hauríem de poder executar:
### Observeu que /$lilyprefix/bin/lilypond és un guió, que estableix
### un valor per a LD_LIBRARY_PATH : això és crucial
      /$lilyprefix/bin/lilypond -jlily,lily,/mnt/lilyloop,/lilyhome test.ly
```

1.3 Missatges d'error

Poden aparèixer diferents missatges d'error en compilar un fitxer:

Advertiment
: Alguna cosa té un aspecte sospitós. Si estem demanant quelcom fora del comú, entendrem el missatge i podrem ignorar-lo. Tot i així, els advertiments solen indicar que alguna cosa va mal amb el fitxer d'entrada.

Error
: És clar que alguna cosa va malament. El pas actual del processament (anàlisi, interpretació o format visual) es donarà per acabat, però el pas següent se saltarà.

Error fatal
: És clar que alguna cosa va malament, i el LilyPond no pot continuar. Poques vegades passa això. La causa més freqüent són els tipus de lletra mal instal·lats.

Error del Scheme
: Els errors que ocorren en executar el codi del Scheme s'intercepten per part de l'intèrpret del Scheme. Si s'està executant amb les opcions `-V` o `--verbose` (detallat) aleshores s'imprimeix una traça de crides de la funció ofensiva.

Error de programació
: Hi ha hagut algun tipus d'inconsistència interna. Aquests missatges d'error estan orientats a ajudar als programadors i als depuradors. Normalment es poden ignorar. En ocasions apareixen en quantitats tan grans que poden entorpir la visió d'altres missatges de sortida.

Abort (bolcat de core)
: Això senyala un error de programació seriós que ha causat la interrupció abrupta del programa. Aquests errors es consideren crítics. Si es topa amb un, envieu un informe de fallada.

Si els errors i advertiments es poden lligar a un punt del fitxer d'entrada, els missatges tenen la forma següent:

```
fitxer:línia:columna: missatge
línia d'entrada problemàtica
```

S'insereix un salt de línia a la línia problemàtica per indicar la columna on es va trobar l'error. Per exemple,

```
prova.ly:2:19: error: no és una duració: 5
```

```
{ c'4 e'
        5 g' }
```

Aquestes posicions són la millor suposició del LilyPond sobre on s'ha produït el missatge d'error, però (per la seva pròpia naturalesa) els advertiment i errors es produeixen quan passa quelcom inesperat. Si no veieu un error a la línia que s'indica del fitxer d'entrada, intenteu comprovar una o dues línies per sobre de la posició indicada.

S'ofereix més informació sobre els errors a la secció Secció 1.4 [Errors comuns], pàgina 12.

1.4 Errors comuns

Les condicions d'error que es descriuen més a sota es produeixen amb freqüència, tot i que la causa no és òbvia o fàcil de trobar. Un cop se han vist i comprès, es gestionen sense problema.

La música se surt de la pàgina

La música que se surt de la pàgina pel marge dret o que apareix exageradament comprimida està causada gairebé sempre per haver introduït una duració incorrecta per a una nota, produint que la nota final d'un compàs s'estengui més enllà de la línia divisòria. Això no és invàlid si la nota final d'un compàs no acaba sobre la línia divisòria introduïda automàticament, atès que simplement se suposa que la nota se solapa a sobre del compàs següent. Però si es produeix una seqüència llarga d'aquestes notes solapades, la música pot aparèixer comprimida o sortir-se de la pàgina perquè els salts de línia automàtiques solament se poden inserir al final dels compassos complets, és a dir, els compassos en els quals totes les notes acaben abans o just al final del compàs.

> **Nota:** Una duració incorrecta pot fer que s'inhibeixin els salts de línia, el que portaria a una sola línia de música molt comprimida o que se surti de la pàgina.
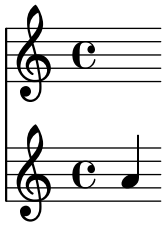
La duració incorrecta es pot trobar fàcilment si s'utilitzen comprovacions de compàs, vegeu Secció "Comprovació de compàs i de número de compàs" in *Referència de la notació*.

Si realment volem tenir una sèrie d'aquests compassos amb notes solapades, hem d'inserir una línia divisòria invisible on volem el salt de línia. Per veure més detalls, consulteu Secció "Barres de compàs" in *Referència de la notació*.

Apareix un pentagrama de més
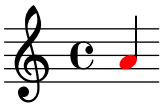
Si no es creen els contextos explícitament amb `\new` o amb `\context`, es crearan discretament tan aviat com es trobi una instrucció que no es pot aplicar a un context existent. A partitures senzilles, la creació automàtica dels contextos és útil, i gairebé tots els exemples dels manuals del LilyPond s'aprofiten d'aquesta simplificació. Però ocasionalment la creació discreta de contextos pot fer aflorar pentagrames o partitures nous o inesperats. Per exemple, podria esperar-se que el codi següent fet que totes les notes dins del pentagrama següent estiguessin acolorides de vermell, però de fet el resultat són dos pentagrames, romanent el de sota amb les notes amb el color negre predeterminat.

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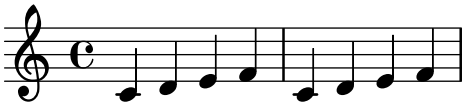


Això és així perquè no hi ha cap context `Staff` quan es processa la instrucció override de sobreescriptura, es crea un implícitament i la sobreescriptura s'aplica a aquest context, però aleshores la instrucció `\new Staff` crea un pentagrama nou i diferent, en el qual es col·loquen les notes. El codi correcte per acolorir totes les notes de vermell és

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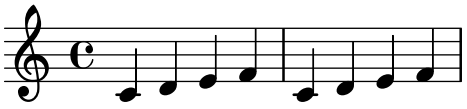


Com a segon exemple, si una instrucció `\relative` s'escriu dins d'una instrucció `\repeat`, el resultat són dos pentagrames, el segon desplaçat respecte al primer, perquè la instrucció `\repeat` genera dos blocs `\relative`, cada un dels quals crea implícitament blocs `Staff` i `Voice`.

```
\repeat unfold 2 {
  \relative { c'4 d e f }
}
```



El problema es resol instanciant el context `Voice` explícitament:

```
\new Voice {
  \repeat unfold 2 {
    \relative { c'4 d e f }
  }
}
```



Missatge d'error Unbound variable %

Aquest missatge d'error apareix al final dels missatges de la consola o del fitxer de registre junt a un missatge "GUILE ha senyalat un error . . ." cada cop que es cridi a una rutina del Scheme que (incorrectament) contingui un comentari *del LilyPond* enlloc d'un comentari *del Scheme*.

Els comentaris del LilyPond comencen amb un símbol de percentatge, (`%`), i no s'han d'utilitzar dins de les rutines del Scheme. Els comentaris del Scheme comencen amb punt i coma, (`;`).

Missatge d'error FT_Get_Glyph_Name

Aquest missatge d'error apareix a la sortida de la consola o al fitxer log de registre si un fitxer d'entrada conté un caràcter que no és ASCII i no s'ha desat en la codificació de caràcters UTF-8. Per veure més detalls, consulteu Secció "Codificació del text" in *Referència de la notació*.

Advertiment sobre que les afinitats del pentagrama sols han de decrèixer

Aquest advertiment pot aparèixer si no hi ha cap pentagrama a la sortida impresa, per exemple si sols hi ha un context `ChordName` i un context `Lyrics` com a un full guia d'acords. Els missatges d'advertiment es poden evitar fent que un dels contextos es comporti com un pentagrama, inserint

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

al començament. Per veure més detalls, consulteu "Espaiat de les línies que no són pautes" a Secció "Espaiat vertical flexible dins dels sistemes" in *Referència de la notació*.

Missatge d'error Unexpected new \new

Un bloc `\score` ha de contenir una *única* expressió musical. Si en comptes d'això conté diverses instruccions `\new Staff`, `\new StaffGroup` o contextos semblants introduïts amb `\new` sense que s'hagin tancat entre claus, `{ ... }`, o dobles parèntesis en angle, `<< ... >>`, així:
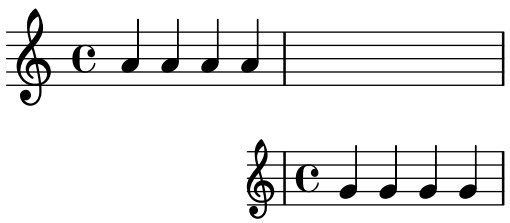
```
\score {
  % Invàlid! Genera error: error de sintaxi, \new inesperat
  \new Staff { ... }
  \new Staff { ... }
}
```

aleshores es produirà un missatge d'error.

Per evitar l'error, tanqueu totes les instruccions `\new` dins de les claus o dobles parèntesis d'angle.

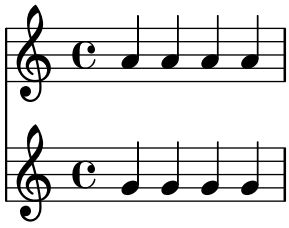
L'ús de claus introdueix les instruccions `\new` de forma seqüencial:

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



però és més probable que us trobeu utilitzant angles dobles de manera que els pentagrames nous s'insereixin en paral·lel, és a dir, simultàniament:

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



2 Actualització de fitxers amb `convert-ly`

La sintaxi del llenguatge d'entrada del LilyPond es modifica de forma habitual per a simplificar-la o millorar-la de diferents maneres. Com a efecte secundari, l'intèrpret del LilyPond sovint ja no és compatible amb els fitxers d'entrada antics. Per posar remei a això es pot utilitzar el programa `convert-ly` per actualitzar fitxers a versions més noves del LilyPond.

2.1 Perquè canvia la sintaxi?

La sintaxi de l'entrada del LilyPond canvia de manera ocasional. A mesura que el propi LilyPond millora, la sintaxi (el llenguatge de l'entrada) es modifica en consonància. A vegades aquests canvis es fan per aconseguir que l'entrada sigui més fàcil de llegir i escriure, i d'altres vegades aquests canvis són per donar cabuda a noves funcionalitats del LilyPond.

Per exemple, se suposa que tots els noms de les propietats de `\paper` i de `\layout` estan escrits sota la norma `primer-segon-tercer`. Tot i així, a la versió 2.11.60, observem que la propietat `printallheaders` no seguia aquesta convenció. Hauríem de deixar-la tal com està (confonent als nous usuaris que han de tractar amb un format d'entrada inconsistent), o canviar-la (empipant als usuaris amb experiència que tenen partitures antigues)? En aquest cas, vam decidir canviar el nom a `print-all-headers`. Afortunadament, aquest canvi es pot automatitzar amb la nostra eina `convert-ly`.

Tanmateix, lamentablement `convert-ly` no pot tractar tots els canvis d'entrada. Per exemple, a la versió 2.4 i anteriors de LilyPond els accents i les lletres no angleses s'introdueixen utilitzant el LaTeX: per exemple `No\"el` (que significa 'Nadal' en francès). Al LilyPond 2.6 i següents el caràcter especial `ë` s'ha d'introduir directament al fitxer del LilyPond com un caràcter UTF-8. `convert-ly` no pot canviar tots els caràcters especials del LaTeX a caràcters de UTF-8: haureu d'actualitzar manualment els vostres fitxers del LilyPond antics.

Les regles de conversió de `convert-ly` funcionen usant correspondència i substitució de patrons de text enlloc d'una comprensió profunda de la sintaxi del LilyPond. Això té diverses conseqüències:

- El bon funcionament de la conversió depèn de la qualitat de cada conjunt de regles que s'apliquen i de la complexitat del canvi corresponent. A vegades les conversions poden necessitar correccions manuals, per la qual cosa la versió antiga hauria de conservar-se a efectes de comparació.
- Solament són possibles les conversions de formats més nous: no hi ha cap conjunt de regles per a la desactualització. Així doncs, la còpia principal de treball d'un fitxer del LilyPond solament s'ha d'actualitzar quan ja no hi ha necessitat de seguir mantenint versions antigues del LilyPond. Els sistemes de control de versions com ara el Git poden ser de gran ajuda per realitzar el manteniment de diverses versions dels mateixos fitxers.
- Els propis programes del LilyPond i de l'Scheme són força robustos enfront als espais afegits i suprimits de manera "creativa", però les regles utilitzades per `convert-ly` tendeixen a fer certes suposicions d'estil. El millor que pot fer-se és seguir l'estil que s'usa als manuals per fer actualitzacions indolores, especialment perquè els propis manuals s'actualitzen usant `convert-ly`.

2.2 Invocació de `convert-ly`

`convert-ly` utilitza el enunciats `\version` dels fitxers d'entrada per detectar el número de versió antic. En gairebé tots els casos, per actualitzar el fitxer d'entrada sols cal executar

```
convert-ly -e elmeufitxer.ly
```

dins del directori que conté el fitxer. Amb això s'actualitza `elmeufitxer.ly` *in situ* i es preserva el fitxer original `elmeufitxer.ly~`.

> **Nota:** convert-ly sempre converteix fins l'últim canvi de sintaxi que és capaç de gestionar. Això significa que el número de \version que apareix al fitxer convertit sol ser inferior al número de versió del propi programa convert-ly.

Per convertir d'un cop tots els fitxers d'entrada que hi ha a un directori, useu

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

De forma alternativa, si volem especificar un nom diferent per al fitxer actualitzar, preservant el fitxer original amb el mateix nom, feu

```
convert-ly elmeufitxer.ly > elmeunoufitxer.ly
```

El programa imprimeix una relació dels números de versió per als que s'han fet conversions. Si no s'imprimeix cap número de versió, el fitxer ja està actualitzat.

Els usuaris del MacOS X poden executar aquesta instrucció sota el menú Compilar > Actualitzar sintaxi.

Els usuaris del Windows han d'introduir aquesta instrucció a una nova ventana del terminal del sistema, que es troba en general sota Inici > Accessoris > Símbol del sistema.

2.3 Opcions de la línia d'ordres per a convert-ly

En general, el programa s'invoca de la manera següent:

```
convert-ly [opció]... fitxer...
```

Es poden donar les opcions següents:

`-d, --diff-version-update`
: Incrementa la cadena \version solament si el fitxer efectivament ha canviat. En tal cas, la capçalera de versió correspondrà a la versió següent a l'últim canvi efectiu. Sense aquesta opció la versió reflecteix l'última conversió que es *va intentar* fer.

`-e, --edit`
: Aplica les conversions directament al fitxer d'entrada, modificant-lo in situ. El fitxer original es canvia de nom a elmeufitxer.ly~. Aquest fitxer de còpia de seguretat podria ser un fitxer ocult en alguns sistemes operatius.

`-b, --backup-numbered`
: Quan s'usa amb l'opció '-e', numera els fitxers de còpia de seguretat de forma que no se sobreescrigui cap versió anterior. Els fitxers de còpia de seguretat podrien ser fitxer ocults en alguns sistemes operatius.

`-f, --from=`*versió_d_origen*
: Estableix la versió des de la qual s'ha de convertir. Si no apareix aquesta opció convert-ly intentarà endevinar-la, bastant-se en la instrucció \version del fitxer. Exemple: --from=2.10.25

`-h, --help`
: Imprimeix l'ajuda d'utilització.

`-l` *nivellderegistre*`, --loglevel=`*nivellderegistre*
: Fixa el grau en el qual la sortida és detallada a *nivellderegistre*. Els valors possibles són NONE (cap), ERROR (errors), WARNING (advertiments), PROGRESS (avenç;predeterminat) i DEBUG (depuració).

`-n, --no-version`
: Normalment convert-ly afegeix un indicador \version a la sortida. L'especificació d'aquesta opció el suprimeix.

`-s, --show-rules`
: Mostra totes les conversions conegudes i surt.

`-t, --to=`*versió_final*

Fixa explícitament a quina `\version` convertir, en cas contrari el valor predeterminat és la versió més actual. Ha de ser més alta que la versió de partida.

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

Per actualitzar fragments del LilyPond en fitxer de texinfo, useu

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

Per veure els canvis en la sintaxi del LilyPond entre dues versions donades, useu

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

2.4 Problemes amb convert-ly

En executar convert-ly a una finestra del Símbol de Sistema sota el Windows sobre un fitxer que té espais al nom o la ruta, és necessari tancar tot el nom del fitxer d'entrada amb tres (!) parelles de cometes:

```
convert-ly """D:/Les meves partitures/Oda.ly""" > "D:/Les meves partitures/nova Oda.ly"
```

Si l'ordre simple `convert-ly -e *.ly` no funciona perquè la instrucció expandida es fa massa llarga, en comptes de fer això l'ordre `convert-ly` es pot posar dins d'un bucle. Aquest exemple per a UNIX actualitza tots els documents `.ly` del directori actual

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

A la finestra del terminal d'ordres del Windows, la instrucció corresponent és

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

No es gestionen tots els canvis al llenguatge. Sols es pot especificar una opció de sortida. L'actualització automàtica del Scheme i les interfícies Scheme del LilyPond és força improbable; prepareu-vos per manipular el codi del Scheme a mà.

2.5 Conversions manuals

En teoria, un programa com `convert-ly` hauria de poder tractar qualsevol canvi de sintaxi. Després de tot, un programa d'ordinador interpreta les versions antiga i nova, per la qual cosa un altre programa d'ordinador podria traduir un fitxer a l'altre[1].

Tot i així, el projecte LilyPond compta amb uns recursos limitats: no totes les conversions s'efectuen automàticament. A continuació hi ha una llista de problemes coneguts.

```
1.6->2.0:
No sempre converteix el baix xifrat correctament, específicament
coses com ara {<
>}.  El comentari de Mats sobre com solucionar el
problema:
   Per poder executar convert-ly
   sobre ell, primer vaig sustituir totes les aparicions de '{<' a quelcom mut com ara '{#'
   i de forma semblant vaig sustituir '>}' amb '&}'.  Després de la conversió, vaig poder
   tornar a canviar-los de '{ #' a '{ <' i de '& }' a '> }'.
 No converteix tot l'etiquetatge de text correctament.  En sintaxi antiga,
 es podien agrupar diverses etiquetes entre parèntesis, per exemple
   -#'((bold italic) "cadena")
   Això es converteix incorrectament a
   -\markup{{\bold italic} "cadena"}
   en comptes del correcte
```

[1] Almenys això és possible en qualsevol fitxer del LilyPond que no contingui Scheme. Si hi ha Scheme dins del fitxer, conté un llenguatge Turing-complet, i ens trobem amb el famós "Problema de l'aturada" informàtica.

```
   -\markup{\bold \italic "cadena"}
2.0->2.2:
 No gestiona \partCombine
 No va \addlyrics => \lyricsto, això trenca algunes partitures amb diverses estrofes
2.0->2.4:
 \magnify no es canvia per \fontsize.
    - \magnify #m => \fontsize #f, on f = 6ln(m)/ln(2)
 remove-tag no es canvia.
    - \applyMusic #(remove-tag '. . .) => \keepWithTag #'. . .
 first-page-number no es canvia.
    - first-page-number no => print-first-page-number = ##f
 Els salts de línia a les cadenes de capçalera no es converteixen.
    - \\\\  com salt de línia a les cadenes de \header  => \markup \center-align <
      "Primera línia" "Segona línia" >
 Els terminadors de crescendo i descrecendo no es converteixen.
    - \rced => \!
    - \rc => \!
2.2->2.4:
 \turnOff (usat a \set Staff.VoltaBracket = \turnOff) no es converteix
adequadament.
2.4.2->2.5.9
 \markup{ \center-align <{ ... }> } s'hauria de convertir a:
 \markup{ \center-align {\line { ... }} }
 però ara, falta el \line.
2.4->2.6
 Els caràcters especials del LaTeX com $~$ al text no es converteixen a UTF8.
2.8
 \score{} ara ha de començar amb una expressió musical.  Qualsevol alta cosa
 (en particular, \header{}) ha d'anar després de la música.
```

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], pàgina ⟨undefined⟩. If you have trouble running `lilypond-book` on Windows or Mac OS X using the command line, then see either Secció "Windows" in *Informació general* or Secció "MacOS X" in *Informació general*.

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'2 e2 \tuplet 3/2 { 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-and-boink.ly}

(If needed, replace @file{screech-and-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) 2.21.4
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)
```
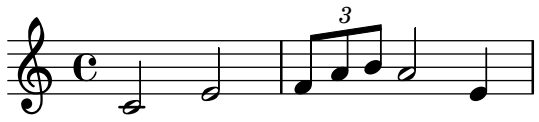
Running `lilypond-book` and `latex` creates a lot of temporary files, which would clutter up the working directory. To remedy this, use the `--output=dir` option. It will create the files in a separate subdirectory `dir`.

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.

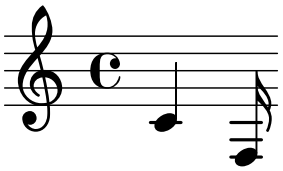
Output

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

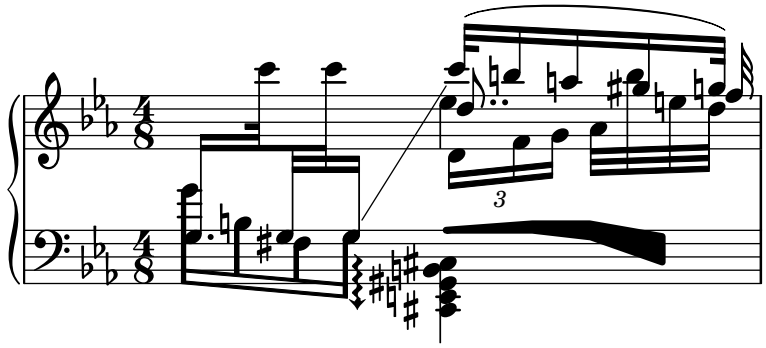


Options are put in brackets.

```
c'4 f16
```



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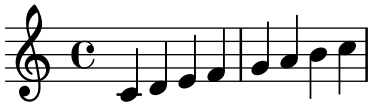


If a tagline is required, either default or custom, then the entire snippet must be enclosed in a \book { } construct.

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

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

A scale in LilyPond



Music engraving by LilyPond 2.21.4—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* (`http://www.ctan.org/tex-archive/info/lshort/english/`) for an overview on how to use LaTeX.

`lilypond-book` provides the following commands and environments to include music in LaTeX files:

- the `\lilypond{...}` command, where you can directly enter short lilypond code
- the `\begin{lilypond}...\end{lilypond}` environment, where you can directly enter longer lilypond code
- the `\lilypondfile{...}` command to insert a lilypond file
- the `\musicxmlfile{...}` command to insert a MusicXML file, which will be processed by `musicxml2ly` and `lilypond`.

In the input file, music is specified with any of the following commands:

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

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

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

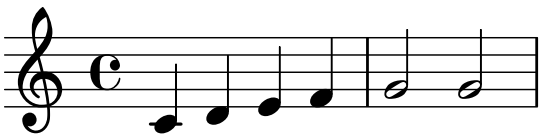
\musicxmlfile[options,go,here]{filename}
```

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`.

Fragments de codi seleccionats

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}
```

Vegeu també

There are specific `lilypond-book` command line options and other details to know when processing LaTeX documents, see Secció 3.4 [Invoking lilypond-book], pàgina 31.

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.

`lilypond-book` provides the following commands and environments to include music into Texinfo files:

- the `@lilypond{...}` command, where you can directly enter short lilypond code
- the `@lilypond...@end lilypond` environment, where you can directly enter longer lilypond code
- the `@lilypondfile{...}` command to insert a lilypond file
- the `@musicxmlfile{...}` command to insert a MusicXML file, which will be processed by `musicxml2ly` and `lilypond`.

In the input file, music is specified with any of the following commands

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

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

@lilypondfile[options,go,here]{filename}

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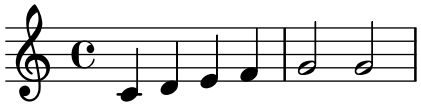
Additionally, `@lilypondversion` displays the current version of lilypond.

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

lilypond-book provides the following commands and environments to include music in HTML files:

- the <lilypond ... /> command, where you can directly enter short lilypond code
- the <lilyond>...</lilypond> environment, where you can directly enter longer lilypond code
- the <lilypondfile>...</lilypondfile> command to insert a lilypond file
- the <musicxmlfile>...</musicxmlfile> command to insert a MusicXML file, which will be processed by musicxml2ly and lilypond.

In the input file, music is specified with any of the following commands:

```
<lilypond options go here>
  YOUR LILYPOND CODE
</lilypond>
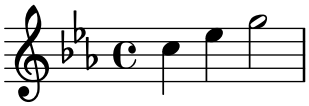
<lilypond options go here: YOUR LILYPOND CODE />

<lilypondfile options go here>filename</lilypondfile>

<musicxmlfile options go here>filename</musicxmlfile>
```

For example, you can write

```
<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>
```

<musicxmlfile> uses the same syntax as <lilypondfile>, but simply references a MusicXML file rather than a LilyPond file.

For a list of options to use with the lilypond or lilypondfile tags, see Secció 3.3 [Music fragment options], pàgina 28.

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-rig
\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 (`http://dblatex.sourceforge.net`), 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. Single-line snippets will always be typeset by default as ragged-right, unless `noragged-right` is explicitly given.

`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 { % begin verbatim
  c'4 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 (but outside the `example` environment generated by a `quote` option).

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`.

`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`:

```
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.

Environments such as;

```
\begin{lilypond} ... \end{lilypond}
```

are not interpreted by LaTeX. Instead, `lilypond-book` extracts those 'environments' into files of its own and runs LilyPond on them. It then takes the resulting graphics and creates a `.tex` file where the `\begin{lilypond}...\end{lilypond}` macros are then replaced by 'graphics inclusion' commands. It is at this time that LaTeX is run (although LaTeX will have run previously, it will have been, effectively, on an 'empty' document in order to calculate things like `\linewidth`).

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

Advertiments i problemes coneguts

The \pageBreak command will not work within a \begin{lilypond} ... \end{lilypond} environment.

Many \paper block variables will also not work within a \begin{lilypond} ... \end{lilypond} environment. Use \newcommand with \betweenLilyPondSystem in the preamble;

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

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. By default, texi2pdf uses pdftex for processing, which you can verify in the console output. In this case, run lilypond-book with the --pdf option so that it creates .pdf snippets instead of .eps files. pdftex is unable to include the latter ones and will output an error message otherwise.

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 Secció 3.5 [Filename extensions], pàgina 34. 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.

-l *loglevel*
--loglevel=*loglevel*
: Set the output verbosity to *loglevel*. Possible values are NONE, ERROR, WARNING, PROGRESS (default) and DEBUG. If this option is not used, and the environment variable LILYPOND_BOOK_LOGLEVEL is set, its value is used as the loglevel.

-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.

`--lily-loglevel=`*loglevel*
: Set the output verbosity of the invoked `lilypond` calls to *loglevel*. Possible values are `NONE`, `ERROR`, `WARNING`, `BASIC_PROGRESS`, `PROGRESS`, `INFO` (default) and `DEBUG`. If this option is not used, and the environment variable `LILYPOND_LOGLEVEL` is set, its value is used as the loglevel.

`--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.

`--redirect-lilypond-output`
: By default, output is displayed on the terminal. This option redirects all output to log files in the same directory as the source files.

`--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. This is equivalent to `--loglevel=DEBUG`.

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

Advertiments i problemes coneguts

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 Secció 3.4 [Invoking lilypond-book], pàgina 31. 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 Capítol 3 [lilypond-book], pàgina 20.

3.6.1 LaTeX

You can include LilyPond fragments in a LaTeX document.

```
\documentclass[]{article}

\begin{document}

Normal LaTeX text.

\begin{lilypond}
\relative {
  a'4 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
@ifnottex
@node Top
@top
@end ifnottex

Texinfo text

@lilypond
\relative {
  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 {
  a'4 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))
             (outfilename (format #f "~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 Secció 4.4 [LilyPond output in other programs], pàgina 46.

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.

4.1.1 Configuring the system

When this functionality is active, LilyPond adds hyperlinks to PDF and SVG files. These hyperlinks are sent to a 'URI helper' or a 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.

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 looks up the environment variables `EDITOR` and `LYEDITOR` to find out and launch the favorite editor to use. `LYEDITOR` will have priority over `EDITOR`, so we recommend using the former especially if you want to use one editor in the terminal and another editor for LilyPond point and click.

Every editor may have a different syntax to open a file in a specific line and column. For user's convenience, LilyPond comes with ready commands for several editors, listed in `scm/editor.scm`. This means that you can simply write the editor binary name, e.g.:

```
export LYEDITOR=atom
```

and this will invoke

```
atom %(file)s:%(line)s:%(column)s
```

where `%(file)s`, `%(line)s` and `%(column)s` are replaced with the file, line and column respectively.

In order to use an editor not listed in `scm/editor.scm`, you should find its specific syntax and assign the full command to `LYEDITOR`. Here's an example for Visual Studio Code editor:

```
export LYEDITOR="code --goto %(file)s:%(line)s:%(column)s"
```

> **Nota:** If you choose Emacs, an extra configuration is needed. You should add the line `(server-start)` to your `~/.emacs` file, otherwise every click on an object in the PDF will open a new Emacs window.

Using Xpdf

For Xpdf on UNIX, the following should be present in `xpdfrc`. On UNIX, this file is found either in `/etc/xpdfrc` or as `$HOME/.xpdfrc`.

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

If you are using Ubuntu, it is likely that the version of Xpdf installed with your system crashes on every PDF file: this state has been persisting for several years and is due to library mismatches. Your best bet is to install a current '`xpdf`' package and the corresponding '`libpoppler`' package from Debian instead. Once you have tested that this works, you might want to use

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

in order to keep Ubuntu from overwriting it with the next 'update' of its crashing package.

Using GNOME 2

For using GNOME 2 (and PDF viewers integrated with it), the magic invocation for telling the system about the 'textedit:' URI is;

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

After that invocation;

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

should call `lilypond-invoke-editor` for opening files.

Using GNOME 3

In GNOME 3, URIs are handled by the 'gvfs' layer rather than by 'gconf'. Create a file in a local directory such as `/tmp` that is called `lilypond-invoke-editor.desktop` and has the contents;

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

and then execute the commands

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

After that invocation;

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

should call `lilypond-invoke-editor` for opening files.

Extra configuration for Evince

If `gnome-open` works, but Evince still refuses to open point and click links due to denied permissions, you might need to change the Apparmor profile of Evince which controls the kind of actions Evince is allowed to perform.

For Ubuntu, the process is to edit the file `/etc/apparmor.d/local/usr.bin.evince` and append the following lines:

```
# For Textedit links
/usr/local/bin/lilypond-invoke-editor Cx -> sanitized_helper,
```

After adding these lines, call

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

Now Evince should be able to open point and click links. It is likely that similar configurations will work for other viewers.

Enabling point and click

Point and click functionality is enabled by default when creating PDF or SVG files.

The point and click links enlarge the output files significantly. For reducing the size of these (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.

Selective point-and-click

For some interactive applications, it may be desirable to only include certain point-and-click items. For example, if somebody wanted to create an application which played audio or video starting from a particular note, it would be awkward if clicking on the note produced the point-and-click location for an accidental or slur which occurred over that note.

This may be controlled by indicating which events to include:

- Hard-coded in the `.ly` file:

```
\pointAndClickTypes #'note-event
\relative {
  c'2\f( f)
}
```

or

```
#(ly:set-option 'point-and-click 'note-event)
\relative {
  c'2\f( f)
}
```

- Command-line:

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

Multiple events can be included:

- Hard-coded in the `.ly` file:

```
\pointAndClickTypes #'(note-event dynamic-event)
\relative {
  c'2\f( f)
}
```

or

```
#(ly:set-option 'point-and-click '(note-event dynamic-event))
\relative {
  c'2\f( f)
}
```

- Command-line:

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

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 (`http://www.vim.org`), 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
syntax on
```

If LilyPond is not installed in the `/usr/local/` directory, change the path appropriately. This topic is discussed in Secció "Other sources of information" in *Manual d'aprenentatge*.

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 Secció "Easier editing" in *Informació general*.

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 (`http://lilypond.org`) for more details.

These are separate programs from `lilypond` itself, and are run on the command line; see ⟨undefined⟩ [Command-line usage], pàgina ⟨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 Secció "MacOS X" in *Informació general*.

Advertiments i problemes coneguts

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 Secció "Staff" in *Referència de funcionament intern* and channels into Secció "Voice" in *Referència de funcionament intern* 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 Secció 4.3 [Converting from other formats], pàgina 42, 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.

Advertiments i problemes coneguts

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 (`http://www.musicxml.org/`) is an XML dialect for representing music notation.

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

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

Note that by 'command-line', we mean the command line of the operating system. See Secció 4.3 [Converting from other formats], pàgina 42, for more information about this.

If `-` is used instead of *file.xml*, `musicxml2ly` reads all input directly from the command line.

The following options are supported by `musicxml2ly`:

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

`--fb --fretboards`
: converts `<frame>` events to a separate FretBoard voice instead of markups.

`-h, --help`
: print usage and a summary of all the available command line options.

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

`--loglevel=LOGLEVEL`
: Sets the output verbosity to *LOGLEVEL*. Possible values are `NONE`, `ERROR`, `WARNING`, `PROGRESS` (default) and `DEBUG`.

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

`-m, --midi`
: activate the midi block in the *.ly* file.

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

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

`--nrp, --no-rest-positions`
: do not convert exact vertical position of rests.

`--nsb, --no-system-breaks`
: ignore system breaks.

`--npl, --no-page-layout`
: do not convert the exact page layout and breaks (shortcut for `--nsb --npb --npm` options).

`--npb, --no-page-breaks`
: ignore page breaks.

`--npm, --no-page-margins`
: ignore page margins.

`--nsd, --no-stem-directions`
: ignore stem directions from MusicXML, use lilypond's automatic stemming instead.

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

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

`--transpose=TOPITCH`
: the interval between pitch `c` and *TOPITCH* to transpose by.

`--sm, --shift-meter=BEATS/BEATTYPE`
: change the length|duration of notes as a function of a given time signature to make the score look faster or slower, (e.g. `4/4` or `2/2`).

`--tc, --tab-clef=TABCLEFNAME`
: switch between two versions of tab clefs (`tab` and `moderntab`).

`--sn --string-numbers=t[rue]/f[alse]`
: deactivate string number stencil with `--string-numbers false`. Default is `true`.

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

`--version`
: show version number and exit.

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

4.3.3 Invoking `abc2ly`

> **Nota:** This is not currently supported and may eventually be removed 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. For example;

```
%%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.

Advertiments i problemes coneguts

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 is not currently supported and may eventually be removed 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 Secció 4.3 [Converting from other formats], pàgina 42, 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

Advertiments i problemes coneguts

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 Secció "Easier editing" in *Informació general*.

4.4 LilyPond output in other programs

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

4.4.1 LuaTex

As well as `lilypond-book` to integrate LilyPond output, there is an alternative program that can be used when using LuaTex called lyluatex (`https://github.com/jperon/lyluatex/blob/master/README.en.md`).

4.4.2 OpenOffice and LibreOffice

LilyPond notation can be added to OpenOffice.org and LibreOffice with OOoLilyPond (`https://github.com/openlilylib/LO-ly`), an OpenOffice.org extension that converts LilyPond files into images within OpenOffice.org documents. OoLilyPond (OLy) works with

recent versions of LibreOffice and OpenOffice. Older versions should work as well. It has even been tested with OpenOffice 2.4 without issues.

4.4.3 Other programs

Other programs that can handle `PNG`, `EPS`, or `PDF` formats should use `lilypond` instead of `lilypond-book`. Each LilyPond output file must be created and inserted separately. Consult the program's own documentation on how to insert files from other sources.

To help 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
}

... music ...
```

To produce `EPS` images;

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

To produce `PNG` images;

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

For transparent `PNG` images

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

If you need to quote many fragments from a large score, you can also use the clip systems feature, see Secció "Extracting fragments of music" in *Referència de la notació*.

4.5 Independent `includes`

Some users have produced files that can be `\included` with LilyPond to produce certain effects and those listed below are part of the LilyPond distribution. Also see Secció "Working with input files" in *Referència de la notació*.

4.5.1 MIDI articulation

The Articulate (`http://www.nicta.com.au/articulate`) project is an attempt to enhance LilyPond's MIDI output and works by adjusting note lengths (that are not under slurs) according to the articulation markings attached to them. For example, a 'staccato' halves the note value, 'tenuto' gives a note its full duration and so on. See Secció "Enhancing MIDI output" in *Referència de la notació*.

5 Suggeriments per escriure fitxers d'entrada

En aquest moment teniu la preparació per començar a escriure fitxers del LilyPond més grans – no sols els petits exemples que apareixen en el tutorial, sinó peces completes –. Però, com heu de procedir per fer-lo?

En la mesura que el LilyPond entengui els seus fitxers i produeixi la sortida que preteníeu, realment no importa massa quin aspecte tinguin els vostres fitxers. Tot i així existeixen algunes altres coses a tenir en compte quan s'escriuen fitxers del LilyPond.

- Què passa si feu un error? L'estructura d'un fitxers del LilyPond pot fer que certs errors es facin més fàcils (o més difícils) de trobar.
- Què passa si voleu compartir els vostres fitxers amb altres persones? De fet, i si voleu alterar els vostres propis fitxers després d'alguns anys? Alguns fitxers del LilyPond s'entenen a primera vista; d'altres poden portar-vos una hora d'entendre.
- Què passa si voleu actualitzar el vostre fitxers del LilyPond per poder-lo usar amb una versió més recent del programa?

  La sintaxi de l'entrada es modifica de forma ocasional segons el LilyPond es va perfeccionant. Gairebé tots els canvis es poden fer de forma automàtica amb `convert-ly`, però a alguns altres els caldria una ajuda manual. Els fitxers del LilyPond es poden estructura perquè siguin més fàcils (o més difícils) d'actualitzar.

5.1 Suggeriments de tipus general

Us presentem alguns suggeriments que us poden servir d'ajuda per evitar o corregir problemes:

- **Incloeu els números de `\version` a tots els fitxers**. Adoneu-vos que totes les plantilles contenen informació sobre la versió (`\version`). Us recomanem molt que sempre incloeu la versió, tot i que els vostre fitxer pugui ser molt petit. Des de l'experiència personal us podem dir que força frustrant intentar recordar el número de versió del LilyPond que estàveu fent servir fa uns anys. `convert-ly` requereix que declareu quina versió del LilyPond fèieu servir.
- **Incloeu comprovacions**: Secció "Comprovació de compàs i de número de compàs" in *Referència de la notació*, Secció "Comprovació d'octava" in *Referència de la notació*. Si incloeu comprovacions de tant en tant, en cas que cometeu un error podreu localitzar-lo molt més ràpidament. Amb quina freqüència és 'de tant en tant'? Depèn de la complexitat de la música. Per a una música molt senzilla, potser tan sols una o dues vegades. Per a una música molt complexa, potser a cada compàs.
- **Un compàs per cada línia de text**. Si hi ha quelcom molt complicat, ja sigui a la pròpia música o a la sortida que desitgeu produir, sovint convé escriure un sol compàs per a cada línia. L'estalvi en espai de pantalla que s'obté acumulant nou compassos per línia no paga la pena si després heu de 'depurar' els fitxers.
- **Comenteu els fitxers**. Utilitzeu o bé números de compàs (de tant en tant), o referències a temes musicals ('segon tema dels violins,' 'quarta variació,' etc.). Potser que no us calguin comentaris quan introduïu una peça per primer cop, però si voleu tornar a ella o modificar quelcom al cap de dos o tres anys, i també si li passeu la font a un amic, serà tot un desafiament determinar les seves intencions o de quina menara estava estructurat el fitxer si no li heu afegit els comentaris.
- **Apliqueu marges a les claus**. Molts problemes estan casat per una falta d'equilibri en el nombre de `{` i `}`.
- **Escriviu les duracions explícitament** al començament de les seccions i identificadors. Si especifiqueu `c4 d e` al principi d'una frase (en lloc de sols `c d e`) us podeu estalviar problemes si reelaboreu la música més tard.

- **Separeu els ajustaments** de les definicions musicals. Consulteu Secció "Estalvi de tecleig mitjançant variables i funcions" in *Manual d'aprenentatge* i Secció "Fulls d'estil" in *Manual d'aprenentatge*

5.2 Gravació de música existent

Si esteu introduint música a partir d'una partitura existent (és a dir, gravant un full de música ja imprès),

- Introduïu al LilyPond cada sistema del manuscrit, o còpia física, per separat (però manteniu la pràctica d'escriure un compàs per línia de text), i comproveu cada sistema quan l'hàgiu acabat. Podeu usar les propietats `showLastLength` o `showFirstLength` per accelerar el procés (vegeu Secció "Salts sobre la música corregida" in *Referència de la notació*.
- Definiu `mBreak = { \break }` i inseriu `\mBreak` dins del fitxer d'entrada on el manuscrit tingui un salt de línia. D'aquesta forma us resultarà molt més fàcil comparar la música del LilyPond amb l'original. Quan hageu acabat de revisar la vostra partitura podreu definir `mBreak = { }` per treure tots aquests salts de línia. Així permetreu el LilyPond col·locar els salts on el LilyPond el consideri més oportú.
- En escriure una part per a un instrument transpositor dins d'una variable, es recomana que les notes estiguin envoltades dins de

  ```
  \transpose c altura-natural {...}
  ```

  (on `altura-natural` és l'afinació natural de l'instrument) de forma que la música dins de la variable estigui realment en Do major. Després podem tornar a transportar-les en sentit invers quan s'utilitza la variable, si és necessari, però potser no vulguem fer-lo (per exemple en imprimir una partitura en afinació de concert, en convertir una part de trombó de clau de Sol a clau de Fa, etc.). És menys probable cometre errors als transports si tota la música que està dins de les variables es troba en un to coherent.

  A més a més, feu els transport exclusivament cap o des de Do major. Això significa que a part de Do major, les úniques tonalitats que usarem seran els tons d'afinació dels instruments transpositors: bes per a una trompeta en Si bemoll, aes per a un clarinet en La bemoll, etc.

5.3 Projectes grans

En treballar en projectes grans es fa essencial tenir una estructura clara als fitxers dels LilyPond:

- **Utilitzeu un identificador per a cada veu**, amb un mínim d'estructura dins de la definició. L'estructura de la secció `\score` és la que canviarà amb major probabilitat: per contra, és extremadament improbable que canviï la definició de `violí` a versions noves del LilyPond.

  ```
  violí = \relative {
  g'4 c'8. e16
  }
  ...
  \score {
    \new GrandStaff {
      \new Staff {
        \violí
      }
    }
  }
  ```

- **Separeu els ajustaments de les definicions musicals**. Ja s'ha mencionat amb anterioritat, però per a projectes grans és vital. Potser haurem de canviar la definició de `fdesprésp`, però en aquest cas sols ho haurem de fer un cop, i encara podrem evitar tocar res dins de `violí`.

```
fdesprésp = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violí = \relative {
g'4\fluegop c'8. e16
}
```

5.4 Solució de problemes

Abans o després escriureu un fitxer que el LilyPond no podrà compilar. Els missatges que el LilyPond proporciona poden ajudar-vos a trobar l'error, però en molts casos haureu de portar endavant algun tipus d'investigació per determinar l'origen del problema. Les eines més poderoses per a aquest propòsit son el comentari d'una sola línia (indicat per `%`) i el comentari de bloc (indicat per `%{...%}`). Si no sabeu on és el problema, comenceu convertint seccions grans del fitxer d'entrada en un comentari. Després d'eliminar una secció convertint-la en un comentari, proveu a compilar un fitxer un altre cop. Si funciona, aleshores el problema hauria d'estar a la porció que havíeu eliminat. Si no funciona, continueu eliminant material (transformant-lo en comentaris) fins que tingueu quelcom que funcioni.

En un cas extrem podríeu acabat amb sols

```
\score {
  <<
    % \melodia
    % \armonia
    % \baix
  >>
  \layout{}
}
```

(en altres paraules: un fitxer sense música)

Si passa això, no abandoneu. Traieu el comentari d'una secció petita – diguem-ne la part del baix – i observeu si funciona. Si no és així, transformeu en comentaris tota la música del baix (però deixeu el `\baix` de la secció `\score` no comentat.

```
bajo = \relative {
%{
  c'4 c c c
  d d d d
%}
}
```

Ara comenceu poc a poc traient comentaris a cada cop més fraccions de la part del `baix` fins que trobeu la línia del problema.

Una altra tècnica de depuració molt útil és la construcció de Secció "Exemples mínims" in *Informació general*.

5.5 Make i els Makefiles

Possiblement totes les plataformes on pot executar-se el LilyPond contemplen una possibilitat de programari anomenada `make`. Aquest programa llegeix un fitxer especial anomenat `Makefile` que defineix les relacions de dependència entre els fitxers i quines instruccions necessitem donar al sistema operatiu per produir un fitxer a partir d'un altre. Per exemple, el fitxer de make detallaria com obtenir `balada.pdf` i `balada.midi` a partir de `balada.ly` mitjançant l'execució del LilyPond.

Hi ha ocasions en les quals és una bona idea crear un `Makefile` per al nostre projecte, bé sigui per la nostra pròpia comoditat o com a cortesia per a altres que possiblement tinguin accés als

nostres fitxers font. Això és cert per a projectes molt grans amb molts fitxers d'inclusió i diferents opcions de sortida (per exemple partitura completa, particel·les, partitura del director, reducció per a piano, etc.), o per a projectes que requereixen ordres difícils per muntar-los (com els projectes de `lilypond-book`). La complexitat i flexibilitat dels Mekfiles varia enormement segons les necessitats i l'habilitat dels autors. El programa GNU Make ve instal·lat a les distribucions del GNU/Linux i al MacOS X, i també existeix per al Windows.

Consulteu el **Manual de GNU Make** per veure tots els detalls sobre l'ús de `make`, atès que el segueix a continuació ofereix sols una pinzellada de tot els és capaç de fer.

Les instruccions que defineixen les regles a un fitxer de make difereixen en funció de la plataforma; per exemple, les diferents formes del GNU/Linux i del MacOS usen `bash`, mentre que el Windows usa `cmd`. Observeu que al MacOS C, hem de configurar el sistema perquè faci servir l'interpret d'ordres. A continuació presentem alguns makefiles d'exemple, amb versions tant per al GNU/Linux/MacOS com per al Windows.

El primer exemple és per a una obra orquestral en quatre moviments amb l'estructura de directoris següent:

```
Sinfonia/
|-- MIDI/
|-- Makefile
|-- Notes/
|   |-- cello.ily
|   |-- xifres.ily
|   |-- trompa.ily
|   |-- oboe.ily
|   |-- trioCordes.ily
|   |-- viola.ily
|   |-- violiU.ily
|   `-- violiDos.ily
|-- PDF/
|-- Particelles/
|   |-- sinfonia-cello.ly
|   |-- sinfonia-trompa.ly
|   |-- sinfonia-oboes.ly
|   |-- sinfonia-viola.ly
|   |-- sinfonia-violiU.ly
|   `-- sinfonia-violiDos.ly
|-- Partitures/
|   |-- sinfonia.ly
|   |-- sinfoniaI.ly
|   |-- sinfoniaII.ly
|   |-- sinfoniaIII.ly
|   `-- sinfoniaIV.ly
`-- sinfoniaDefs.ily
```

Els fitxers `.ly` dels directoris `Partitures` i `Particelles` obtenen les notes de fitxers `.ily` que estan al directori `Notes`:

```
%%% principi del fitxer "sinfonia-cello.ly"
\include ../definicionsSinf.ily
\include ../Notes/cello.ily
```

El makefile tindrà els objectius de `partitura` (la peça completa en tot el seu esplendor), `moviments` (partitura completa dels moviments individuals) i `particel·les` (parts individuals per als faristols). També hi ha un objectiu `fitxer` que produeix un fitxer tar de distribució

(tarball) dels fitxers font, adequat per compartir-lo a través de la web o per correu electrònic. A continuació presentem el makefile per a GNU/Linux o MacOS C. S'ha de desar amb el nom exacte `Makefile` al directori superior del projecte:

> **Nota:** Quan es defineix un objectiu o una regla de patró, les línies següents han de començar amb tabuladors, no amb espais.

```
# nom principal dels fitxers de sortida
nom = sinfonia
# determinació del nombre de processadors
CPU_CORES=`cat /proc/cpuinfo | grep -m1 "cpu cores" | sed s/".*: "//`
# L'ordre per executar el LilyPond
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click -djob-count=$(CPU_CORES)

# Els sufixos utilitzats a aquest Makefile.
.SUFFIXES: .ly .ily .pdf .midi

# Els fitxers d'entrada i de sortida es busquen dins dels directoris relacionats a
# la variable VPATH.  Tots ells són subdirectoris del directori
# en curs (donat per la variable de GNU make `CURDIR').
VPATH = \
  $(CURDIR)/Partitures \
  $(CURDIR)/PDF \
  $(CURDIR)/Particelles \
  $(CURDIR)/Notes

# La regla de patró per crear fitxers PDF i MIDI a partir fitxers d'entrada LY
# Els fitxers de sortida .pdf es col·loquen al subdirectori `PDF', i els fitxers
# .midi van al subdirectori `MIDI'.
%.pdf %.midi: %.ly
        $(LILY_CMD) $<; \           # Aquesta línia comença amb un salt de tabulació
        if test -f "$*.pdf"; then \
            mv "$*.pdf" PDF/; \
        fi; \
        if test -f "$*.midi"; then \
            mv "$*.midi" MIDI/; \
        fi

notes = \
  cello.ily \
  trompa.ily \
  oboe.ily \
  viola.ily \
  violiU.ily \
  violiDos.ily

# Dependències dels moviments
$(nom)I.pdf: $(nom)I.ly $(notes)
$(nom)II.pdf: $(nom)II.ly $(notes)
$(nom)III.pdf: $(nom)III.ly $(notes)
$(nom)IV.pdf: $(nom)IV.ly $(notes)
```

```
# Dependències de la partitura completa.
$(nom).pdf: $(nom).ly $(notes)

# Dependències de les particel·les.
$(nom)-cello.pdf: $(nom)-cello.ly cello.ily
$(nom)-trompa.pdf: $(nom)-trompa.ly trompa.ily
$(nom)-oboes.pdf: $(nom)-oboes.ly oboe.ily
$(nom)-viola.pdf: $(nom)-viola.ly viola.ily
$(nom)-violiU.pdf: $(nom)-violiU.ly violiU.ily
$(nom)-violiDos.pdf: $(nom)-violiDos.ly violiDos.ily

# Teclegeu `make partitura' per generar la partitura completa dels quatre
# moviments com un fitxer únic.
.PHONY: partitura
partitura: $(nom).pdf

# Teclegeu `make particelles' per generar totes les particel·les
# Teclegeu `make pepet.pdf' per generar la particel·la de
# l'instrument `pepet'.

# Exemple: `make sinfonia-cello.pdf'.
.PHONY: particellas
particellas: $(nom)-cello.pdf \
       $(nom)-violinUno.pdf \
       $(nom)-violinDos.pdf \
       $(nom)-viola.pdf \
       $(nom)-oboes.pdf \
       $(nom)-trompa.pdf

# Teclegeu `make moviments' per generar els fitxers dels
# quatre moviments de forma separada.
.PHONY: moviments
moviments: $(nom)I.pdf \
           $(nom)II.pdf \
           $(nom)III.pdf \
           $(nom)IV.pdf

all: partitura particelles moviments

fitxer:
        tar -cvvf stamitz.tar \       # aquesta línia comença amb un salt de tabulació
        --exclude=*pdf --exclude=*~ \
        --exclude=*midi --exclude=*.tar \
        ../Stamitz/*
```

A la plataforma Windows hi ha certes complicacions. Després de descarregar i instal·lar el programa GNU Make per al Windows, haurem de configurar la ruta adequada a las variables d'entorn del sistema de que l'intèrpret d'ordres del DOS pugui trobar el programa Make. Per fer-lo, polseu amb el botó dret sobre "El meu ordinador", escolliu `Propietats` i `Avançades`. Polseu sobre `Variables d'entorn`, i després a la pestanya `Variables del sistema`, seleccioneu

`Ruta`, polseu sobre `edita` i afegiu la ruta al fitxer executable de GNU Make, amb la qual cosa quedarà quelcom semblant al següent:

```
C:\Fitxers de programa\GnuWin32\bin
```

El makefile en si s'ha de modificar perquè gestioni diverses instruccions de l'intèrpret d'ordres i perquè pugui tractar amb els espais que apareixen al nom d'alguns directoris del sistema predeterminats. L'objectiu `fitxer` s'elimina perquè el Windows no té l'ordre `tar`, i el Windows a més té una extensió predeterminada diferent per als fitxers MIDI.

```
## VERSIÓ PER AL WINDOWS
##
nom = sinfonia
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click \
                    -djob-count=$(NUMBER_OF_PROCESSORS)

#obtenció del nom 8.3 de CURDIR (truc per als espais a PATH)
workdir = $(shell for /f "tokens=*" %%b in ("$(CURDIR)") \
          do @echo %%~sb)

.SUFFIXES: .ly .ily .pdf .mid

VPATH = \
  $(workdir)/Partitures \
  $(workdir)/PDF \
  $(workdir)/Particelles \
  $(workdir)/Notes

%.pdf %.mid: %.ly
        $(LILY_CMD) $<      # aquesta línia comença amb un salt de tabulació
        if exist "$*.pdf"  move /Y "$*.pdf"  PDF/ # començament amb tab
        if exist "$*.mid" move /Y "$*.mid" MIDI/  # començament amb tab

notes = \
  cello.ily \
  xifres.ily \
  trompa.ily \
  oboe.ily \
  trioCordes.ily \
  viola.ily \
  violiU.ily \
  violiDos.ily

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

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

$(nom)-cello.pdf: $(nom)-cello.ly cello.ily
$(nom)-trompa.pdf: $(nom)-trompa.ly trompa.ily
$(nom)-oboes.pdf: $(nom)-oboes.ly oboe.ily
```

```
$(nom)-viola.pdf: $(nom)-viola.ly viola.ily
$(nom)-violiU.pdf: $(nom)-violiU.ly violiU.ily
$(nom)-violiDos.pdf: $(nom)-violiDos.ly violiDos.ily

.PHONY: partitura
partitura: $(nom).pdf

.PHONY: particelles
particelles: $(nom)-cello.pdf \
       $(nom)-violiU.pdf \
       $(nom)-violiDos.pdf \
       $(nom)-viola.pdf \
       $(nom)-oboes.pdf \
       $(nom)-trompa.pdf

.PHONY: moviments
moviments: $(nom)I.pdf \
           $(nom)II.pdf \
           $(nom)III.pdf \
           $(nom)IV.pdf

all: partitura particelles moviments
```

El Makefile següent és per a un document de `lilypond-book` fet en LaTeX. Aquest projecte té un índex, que requereix executar l'ordre `latex` dues vegades per actualitzar els enllaços. Tots els fitxers de sortida s'emmagatzemen al directori `sortida` per als documents .pdf i al directori `sortidahtml` per a la sortida en format html.

```
SHELL=/bin/sh
NOM=elmeuprojecte
DIR_SORTIDA=sortida
DIR_WEB=sortidahtml
VISUALITZADOR=acroread
NAVEGADOR=firefox
LILYBOOK_PDF=lilypond-book --output=$(DIR_SORTIDA) --pdf $(NOM).lytex
LILYBOOK_HTML=lilypond-book --output=$(DIR_WEB) $(NOM).lytex
PDF=cd $(DIR_SORTIDA) && pdflatex $(NOM)
HTML=cd $(DIR_WEB) && latex2html $(NOM)
INDEX=cd $(DIR_SORTIDA) && makeindex $(NOM)
VISTA_PREVIA=$(VISUALITZADOR) $(DIR_SORTIDA)/$(NOM).pdf &

all: pdf web desar

pdf:
        $(LILYBOOK_PDF)  # comença amb un tab
        $(PDF)           # comença amb un tab
        $(INDEX)        # comença amb un tab
        $(PDF)           # comença amb un tab
        $(VISTA_PREVIA)  # comença amb un tab

web:
        $(LILYBOOK_HTML) # comença amb un tab
        $(HTML)          # comença amb un tab
```

```
        cp -R $(DIR_WEB)/$(NOM)/ ./  #
        $(NAVEGADOR) $(NOM)/$(NOM).html &  # comença amb un tab

desar: pdf
        cp $(DIR_SORTIDA)/$(NOM).pdf $(NOM).pdf  # comença amb un tab

clean:
        rm -rf $(DIR_SORTIDA) # comença amb un tab

web-clean:
        rm -rf $(DIR_WEB) # comença amb un tab

fitxer:
        tar -cvvf elmeuprojecte.tar \ # comença amb un tab
        --exclude=sortida/* \
        --exclude=sortidahtml/* \
        --exclude=elmeuprojecte/* \
        --exclude=*midi \
        --exclude=*pdf \
        --exclude=*~ \
        ../ElMeuProjecte/*
```

PERFER: aconseguir que funcioni a windows

El makefile anterior no funciona al Windows. Una alternativa per als usuaris del Windows seria crear un fitxer de lots senzill que contingui les ordres de muntatge. Això no segueix les dependències com ho fa un makefile, però almenys redueix el procés de construcció a una sola instrucció. Deseu el codi següent com `muntatge.bat` o `muntatge.cmd`. El fitxer de lots es pot executar en la línia d'ordres del DOS o simplement fent doble clic sobre la seva icona.

```
lilypond-book --output=sortida --pdf elmeuprojecte.lytex
cd sortida
pdflatex elmeuprojecte
makeindex elmeuprojecte
pdflatex elmeuprojecte
cd ..
copy sortida\elmeuprojecte.pdf ElMeuProjecte.pdf
```

Vegeu també

Manual d'utilització del programa: Secció "Utilització des de la línia d'ordres" in *Utilització del programa*, Secció "lilypond-book" in *Utilització del programa*

Annex A GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
`https://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 `https://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.

Annex B Índex del LilyPond

A

ABC 45
actualització d'un fitxer del LilyPond 16
actualització de fitxers d'entrada antics 16
advertiment 11
apuntar i clicar, línia d'ordres 4
Articulate project 47

B

`Barres de compàs` 12

C

carpeta, dirigir la sortida cap a 4
cerca, ruta de 2
chroot, executar dins d'una gàbia 2
Coda Technology 46
`Codificació del text` 13
coloring, syntax 41
`Comprovació d'octava` 48
`Comprovació de compàs i de número de compàs` 12, 48
convert-ly 16
crides, traça de 11

D

docbook 20
DocBook, adding music 20
documents, adding music 20
dvips 31

E

`Easier editing` 42, 46
editors 41
emacs 41
`Enhancing MIDI output` 47
enigma 46
Enigma Transport Format 46
error 11
error de programació 11
error del Scheme 11
error fatal 11
error, format dels missatges de 11
error, missatges d'error 11
`Espaiat vertical flexible dins dels sistemes` 14
`Estalvi de tecleig mitjançant variables i funcions` 49
ETF 46
Evince 40
`Exemples mínims` 50
expressions del Scheme, avaluació 2
External programs, generating LilyPond files 46
`Extracting fragments of music` 47

F

fatal, error 11
file size, output 40
Finale 46
fitxers, cerca de 2
format, sortida 2
fragments, music 47
`Fulls d'estil` 49

H

\header in LaTeX documents 25
HTML 20
HTML, adding music 20

I

invocació de `lilypond` 2
invoking dvips 31

L

línia d'ordres, opcions de 2
LANG 9
LaTex 20
LaTeX, adding music 20
LibreOffice.org 46
`lilypond-book` 56
LILYPOND_DATADIR 9
loglevel 3
LuaTex 46
lyluatex 46

M

`MacOS X` 1, 20, 42
make 50
make, fitxers de 50
`Manuals` 1
MIDI 42, 47
missatges d'error 11
modes, editor 41
modificadors 2
music fragments, quoting 47
musicology 20
MusicXML 44

O

OOoLilyPond 46
opcions de la línia d'ordres per a `lilypond` 2
OpenOffice.org 46
`Other sources of information` 42
outline fonts 31

P

PDF (format de document portàtil), sortida de 4
PNG (Portable Network Graphics), sortida 4
point and click 39
Postscript (PS), sortida 4
preview image 27
programació, error de 11
PS (Postscript), sortida 4

Q

quoting, music fragments 47

R

registre, nivell de 3

S

`Salts sobre la música corregida` 49
Scheme, avaluació d'expressions 2
Scheme, error de 11
sortida neta, fixar el nivell 3
sortida, establir el nom del fitxer de 4
sortida, format 2
sortida, PDF (format de document portàtil) 4
sortida, PNG (Portable Network Graphics) 4
sortida, PS (Postscript) 4
`Staff` 43
syntax coloring 41

T

texi 20
texinfo 20
Texinfo, adding music 20
thumbnail 27
titling and lilypond-book 25
titling in HTML 27
traça del Scheme 11
`Tutorial` 1
type1 fonts 31

U

`Utilització des de la línia d'ordres` 56

V

vim 41
`Voice` 43

W

`Windows` 20
`Working with input files` 47

X

Xpdf 39