

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 2.19.18. 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://www.lilypond.org/>.

Copyright © 1999–2014 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 2.19.18

Índex General

1	Execució del LilyPond	1
1.1	Utilització normal	1
1.2	Utilització des de la línia d'ordres	1
	Invocació del <code>lilypond</code>	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	Erroros 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	14
	Advertiment sobre que les afinitats del pentagrama sols han de decreïxer	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 <code>convert-ly</code>	16
2.3	Opcions de la línia d'ordres per a <code>convert-ly</code>	17
2.4	Problemes amb <code>convert-ly</code>	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 L ^A T _E X	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 <code>lilypond-book</code>	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	39
	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	41
	Vim mode	42
	Other editors	42
4.3	Converting from other formats	42
4.3.1	Invoking <code>midi2ly</code>	42
4.3.2	Invoking <code>musicxml2ly</code>	43
4.3.3	Invoking <code>abc2ly</code>	44
4.3.4	Invoking <code>etf2ly</code>	45
4.3.5	Other formats	45
4.4	LilyPond output in other programs	46
	Many quotes from a large score	46
	Inserting LilyPond output into OpenOffice and LibreOffice	46
	Inserting LilyPond output into other programs	46
4.5	Independent <code>includes</code>	46
4.5.1	MIDI articulation	46
5	Suggestions for writing files	48
5.1	General suggestions	48
5.2	Typesetting existing music	49
5.3	Large projects	49
5.4	Troubleshooting	50
5.5	Make and Makefiles	51
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 independent¹.

Si ‘`fitxer.ly`’ conté més d'un bloc `\score`, la resta de les partitures s'obtidran com a sortida en fitxers numerats, començant per ‘`fitxer-1.pdf`’. A més, el valor de `output-suffix` (sufix de sortida) s'insereix 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

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.

¹ L'estat del GUILF 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.

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 dona 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 <i>.tex</i> , <i>.texi</i> , <i>.count</i> al 'back-end' 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.

<code>eps</code>		PostScript encapsulat. Bolca cada pàgina o sistema com un fitxer ‘EPS’ diferent, sense tipus tipogràfics de lletra, i com un fitxer ‘EPS’ enquadrant amb totes les pàgines o sistemes que inclouen els tipus de lletra. Utilitzat com a opció predeterminada per part de <code>lilypond-book</code> .
<code>null</code>		No produeix cap partitura impresa a la sortida; té el mateix efecte que <code>-dno-print-pages</code> .
<code>svg</code>		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 <code>‘/usr/share/lilypond/VERSION/fonts/otf/’</code>) al directori <code>‘~/fonts/’</code> . La sortida SVG hauria de ser compatible amb qualsevol editor o client de SVG. També hi ha una opció <code>svg-woff</code> (vegeu més avall) per usar els fitxers de tipus de lletra woff al ‘rerefons’ SVG.
<code>scm</code>		Bolcat de les instruccions de dibuix internes basades en Scheme, en brut.
<code>check-internal-types n</code>	<code>#f</code>	Comprova el tipus de cada assignació de propietats.
<code>clip-systems</code> (retalla els sistemes de pentagrames)	<code>#f</code>	Genera fragments d’imatge retallats d’una partitura.
<code>datadir</code> (directori de dades)		Prefix dels fitxers de dades (sols lectura).
<code>debug-gc</code>	<code>#f</code>	Bolca estadístiques de depuració de memòria.
<code>debug-gc-assert-parsed-dead</code>	<code>#f</code>	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 <code>~-ddebug-gc</code> .
<code>debug-lexer</code>	<code>#f</code>	Depuració de l’analitzador lèxic flex.
<code>debug-page-breaking-scoring</code>	<code>#f</code>	Bolca les partitures per a moltes configuracions de salts de pàgina diferents.

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

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

<code>log-file</code>	<code>#f</code> <code>[fitxer]</code>	Si es dona a una cadena <code>fitxer</code> como a segon argument, redirigeix la sortida al fitxer de registre <code>fitxer.log</code> .
<code>max-markup-depth</code>	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.
<code>midi-extension</code>	"midi"	Fixa l'extensió de fitxer predeterminat per al fitxer de sortida MIDI a la cadena donada.
<code>music-strings-to-paths</code>	<code>#f</code>	Converteix les cadenes de text a rutes quan els glifs pertanyen a un tipus de lletra de tipografia musical.
<code>paper-size</code>	<code>\ "a4\"</code>	Estableix la mida predeterminada del paper. Observeu que la cadena ha d'anar tancada entre cometes dobles.
<code>pixmap-format</code>	<code>png16m</code>	Fixa el format de sortida del Ghostscript per a les imatges de píxels.
<code>point-and-click</code>	<code>#f</code>	Afegeix enllaços d'apuntar i clicar a la sortida PDF. Vegeu Secció 4.1 [Point and click] , pàgina 39 .
<code>preview</code>	<code>#f</code>	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.

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

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

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.

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

<code>strokeadjust</code>	<code>#f</code>	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.
<code>svg-woff</code>	<code>#f</code>	Usar fitxers de tipus tipogràfic de lletra de woff al rerefons SVG.
<code>trace-memory-frequency</code>	<code>#f</code>	Registra l'ús de cèl·lules del Scheme aquesta quantitat de vegades per segon. Bolca els resultats en <code>FITXER.stacks</code> i en <code>FITXER.graph</code> .
<code>trace-scheme-coverage</code>	<code>#f</code>	Registra la cobertura dels fitxers del Scheme a <code>FITXER.cov</code> .
<code>verbose</code>	<code>#f</code>	Sortida detallada, és a dir el nivell de registre en DEBUT (sols lectura).
<code>warning-as-error</code>	<code>#f</code>	Canvia tots els missatges d'avertiment 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'avertiment.

`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 fitxers 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 ghostscript...
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 sospitos. 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'interpret 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 advertiments 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 automàtics 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 sobreescritura, es crea un implícitament i la sobreescritura 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' { c4 d e f }
}
```



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

```
\new Voice {
  \repeat unfold 2 {
    \relative c' { c4 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 decreïxer

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 claudàtors corbs, `{ ... }`, 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 dels claudàtors corbs o dobles parèntesis d'angle.

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

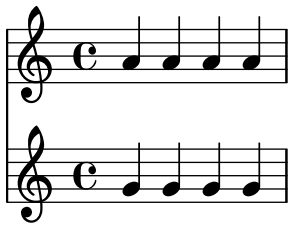
```
\score {
  {
    \new Staff { a' a' a' a' }
    \new Staff { g' g' g' g' }
  }
}
```



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'interpret 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\"e1` (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 enunciat `\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 fitxers 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'altre¹.

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 substituir totes les aparicions de `'{<'` a quelcom mut com ara `'{#'`

i de forma semblant vaig substituir `'>}'` 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 $f = 6\ln(m)/\ln(2)$
 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 \sim 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 L^AT_EX 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 [\[Command-line usage\]](#), 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 L^AT_EX, 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, L^AT_EX, 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 L^AT_EX. The example also contains explanatory text, so we will not comment on it further.

Input

```
\documentclass[a4paper]{article}
```

```
\begin{document}
```

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

```
\begin{lilypond}
\relative c' {
  c2 e2 \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) 2.19.18
```

```
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 L^AT_EX example shown above.¹ This finishes the tutorial section.

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

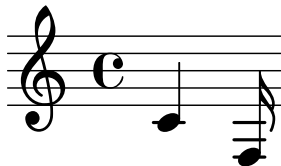
Output

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

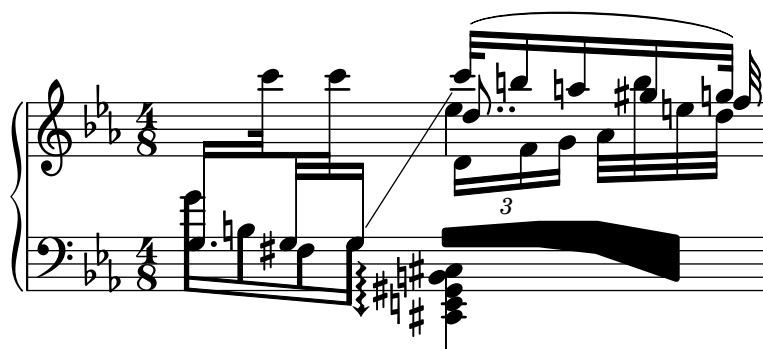


Options are put in brackets.

```
c'4 f16
```



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



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

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

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

A scale in LilyPond



Music engraving by LilyPond 2.19.18—www.lilypond.org

3.2 Integrating music and text

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

3.2.1 L^AT_EX

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

See *The Not So Short Introduction to L^AT_EX* for an overview on how to use L^AT_EX.

lilypond-book provides the following commands and environments to include music in L^AT_EX 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 L^AT_EX.

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 `<lilypond>...</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-right r
\context Staff \with {
  \remove "Time_signature_engraver"
  \remove "Clef_engraver"}
  { c4( fis) }
  </programlisting>
  </textobject>
</inlinemediaobject>
```

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

Processing the DocBook document

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

3.3 Music fragment options

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

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

The following options are available for LilyPond commands:

staffsize=ht

Set staff size to *ht*, which is measured in points.

ragged-right

Produce ragged-right lines with natural spacing, i.e., **ragged-right = ##t** is added to the LilyPond snippet. 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*0.4in and put the output into a quotation block. The value '0.4in' can be controlled with the **exampleindent** option.

exampleindent

Set the amount by which the **quote** option indents a music snippet.

relative

relative=n

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

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

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

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

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

will be printed with a verbatim block like

```
c4 e2 g4
f2 e
```

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

addversion

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

texidoc

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

L^AT_EX

There are two ways of processing your L^AT_EX document for printing or publishing: getting a PDF file directly with PDFL^AT_EX, or getting a PostScript file with L^AT_EX via a DVI to PostScript translator like **dvips**. The first way is simpler and recommended¹, 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 PDFL^AT_EX, use:

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

To produce PDF output via L^AT_EX/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 L^AT_EX. 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

¹ Note that PDFL^AT_EX and L^AT_EX may not be both usable to compile any L^AT_EX document, that is why we explain the two ways.

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`).

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. 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.

```

Advertisements 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
<code>' .html '</code>	HTML
<code>' .htmlly '</code>	HTML
<code>' .itely '</code>	Texinfo
<code>' .latex '</code>	\LaTeX
<code>' .lytex '</code>	\LaTeX
<code>' .lyxml '</code>	DocBook
<code>' .tely '</code>	Texinfo
<code>' .tex '</code>	\LaTeX
<code>' .texi '</code>	Texinfo
<code>' .texinfo '</code>	Texinfo
<code>' .xml '</code>	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 c'' {
```

```
  a4 b c d
```

```
}
```

```
\end{lilypond}
```

More LaTeX text, and options in square brackets.

```
\begin{lilypond}[fragment,relative=2,quote,staffsize=26,verbatim]
```

```
d4 c b a
```

```
\end{lilypond}
\end{document}
```

3.6.2 Texinfo

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

```
\input texinfo @node Top
@top
```

Texinfo text

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

More Texinfo text, and options in brackets.

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

```
@bye
```

3.6.3 html

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

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

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

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

```
</body>
</html>
```

3.6.4 xelatex

```

\documentclass{article}
\usepackage{ifxetex}
\ifxetex
%xetex specific stuff
\usepackage{xunicode,fontspec,xltxtra}
\setmainfont[Numbers=OldStyle]{Times New Roman}
\setsansfont{Arial}
\else
%This can be empty if you are not going to use pdftex
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{mathptmx}%Times
\usepackage{helvet}%Helvetica
\fi
%Here you can insert all packages that pdftex also understands
\usepackage[ngerman,finnish,english]{babel}
\usepackage{graphicx}

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

```

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

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

```

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

```

```

\noindent
and so on.

```

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

```

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

```

```
\end{document}
```

3.7 Sharing the table of contents

These functions already exist in the `OrchestralLily` package:

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

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

Exporting the ToC from LilyPond

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

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

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

Importing the ToC into L^AT_EX

In L^AT_EX, 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@xxxxxxxx}{\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 tests the environment variable `EDITOR` for the following patterns,

```
emacs      this will invoke
            emacsclient --no-wait +line:column file
gvim       this will invoke
            gvim --remote +:line:nomcolumn file
nedit      this will invoke
            nc -noask +line file'
```

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

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

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

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-edit
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' {
  c2\f( f)
}
```

or

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

or

```
 #(ly:set-option 'point-and-click '(note-event dynamic-event))
\relative c' {
  c2\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**, 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](#) for more details.

These are separate programs from `lilypond` itself, and are run on the command line; see [\[Command-line usage\]](#), [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,

`mid2ly [option]... mid-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 `mid2ly`.

- `-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.

Advertisements 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 is an XML dialect for representing music notation.

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

It is invoked from the command-line as follows,

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

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

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

The following options are supported by `musicxml2ly`:

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

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

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

--loglevel=loglevel
Set 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 midi-block.

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

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

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

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

-v, --verbose
be verbose.

--version
print version information.

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

4.3.3 Invoking abc2ly

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

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

<http://www.walshaw.plus.com/abc/learn.html>.

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

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

The following options are supported by abc2ly:

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

-h, --help
this help

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

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

--version
    print version information.
```

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

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

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

Similarly,

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

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

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 program is not supported, and may be remove from future versions of LilyPond.

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

It is invoked from the command-line as follows.

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

Note that by ‘command-line’, we mean the command line of the operating system. See [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`.

Many quotes from a large score

If you need to quote many fragments from a large score, you can also use the clip systems feature, see [Secció “Extracting fragments of music”](#) in *Referència de la notació*.

Inserting LilyPond output into OpenOffice and LibreOffice

LilyPond notation can be added to OpenOffice.org and LibreOffice with [OOoLilyPond](#).

Inserting LilyPond output into other programs

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

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

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

```
{ c1 }
```

To produce useful image files:

EPS

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

PNG

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

A transparent PNG

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

4.5 Independent includes

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

4.5.1 MIDI articulation

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

The *articulate* project is one attempt to get more of the information in the score into MIDI. It works by shortening notes not under slurs, to ‘articulate’ the notes. The amount of shortening depends on any articulation markings attached to a note: staccato halves the note value, tenuto gives a note its full duration, and so on. The script also realises trills and turns, and could be extended to expand other ornaments such as mordents.

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

Advertiments i problemes coneguts

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

5 Suggestions for writing files

Now you're ready to begin writing larger LilyPond input files – not just the little examples in the tutorial, but whole pieces. But how should you go about doing it?

As long as LilyPond can understand your input files and produce the output that you want, it doesn't matter what your input files look like. However, there are a few other things to consider when writing LilyPond input files.

- What if you make a mistake? The structure of a LilyPond file can make certain errors easier (or harder) to find.
- What if you want to share your input files with somebody else? In fact, what if you want to alter your own input files in a few years? Some LilyPond input files are understandable at first glance; others may leave you scratching your head for an hour.
- What if you want to upgrade your LilyPond file for use with a later version of LilyPond? The input syntax changes occasionally as LilyPond improves. Most changes can be done automatically with `convert-ly`, but some changes might require manual assistance. LilyPond input files can be structured in order to be easier (or harder) to update.

5.1 General suggestions

Here are a few suggestions that can help to avoid (and fix) the most common problems when typesetting:

- **Always include a `\version` number in your input files** no matter how small they are. This prevents having to remember which version of LilyPond the file was created with and is especially relevant when `<undefined>` [Updating files with `convert-ly`], pàgina `<undefined>` command (which requires the `\version` statement to be present); or if sending your input files to other users (e.g. when asking for help on the mail lists). Note that all of the LilyPond templates contain `\version` numbers.
- **For each line in your input file, write one bar of music.** This will make debugging any problems in your input files much simpler.
- **Include *Secció “Bar and bar number checks”* in *Referència de la notació* as well as *Secció “Octave checks”* in *Referència de la notació*.** Including ‘checks’ of this type in your input files will help pinpoint mistakes more quickly. How often checks are added will depend on the complexity of the music being typeset. For simple compositions, checks added at a few at strategic points within the music can be enough but for more complex music, with many voices and/or staves, checks may be better placed after every bar.
- **Add comments within input files.** References to musical themes (i.e. ‘second theme in violins’, ‘fourth variation,’ etc.), or simply including bar numbers as comments, will make navigating the input file much simpler especially if something needs to be altered later on or if passing on LilyPond input files to another person.
- **Add explicit note durations at the start of ‘sections’.** For example, `c4 d e f` instead of just `c d e f` can make rearranging the music later on simpler.
- **Learn to indent and align braces and parallel music.** Many problems are often caused by either ‘missing’ braces. Clearly indenting ‘opening’ and ‘closing’ braces (or `<<` and `>>` indicators) will help avoid such problems. For example;

```
\new Staff {
  \relative g' {
    r4 g8 g c8 c4 d |
    e4 r8 |
    % Ossia section
    <<
```

```

        { f8 c c | }
        \new Staff {
            f8 f c |
        }
    >>
    r4 |
}

```

is much easier to follow than;

```

\new Staff { \relative g' { r4 g8 g c4 c8 d | e4 r8
% Ossia section
<< { f8 c c } \new Staff { f8 f c } >> r4 | } }

```

- **Keep music and style separate** by putting overrides in the `\layout` block;

```

\score {
    ...music...
    \layout {
        \override TabStaff.Stemstencil = ##f
    }
}

```

This will not create a new context but it will apply when one is created. Also see [Secció “Saving typing with variables and functions”](#) in *Manual d’aprenentatge*, and [Secció “Style sheets”](#) in *Manual d’aprenentatge*.

5.2 Typesetting existing music

If you are entering music from an existing score (i.e., typesetting a piece of existing sheet music),

- Enter the manuscript (the physical copy of the music) into LilyPond one system at a time (but still only one bar per line of text), and check each system when you finish it. You may use the `showLastLength` or `showFirstLength` properties to speed up processing – see [Secció “Skipping corrected music”](#) in *Referència de la notació*.
- Define `mBreak = { \break }` and insert `\mBreak` in the input file whenever the manuscript has a line break. This makes it much easier to compare the LilyPond music to the original music. When you are finished proofreading your score, you may define `mBreak = { }` to remove all those line breaks. This will allow LilyPond to place line breaks wherever it feels are best.
- When entering a part for a transposing instrument into a variable, it is recommended that the notes are wrapped in

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

(where `natural-pitch` is the open pitch of the instrument) so that the music in the variable is effectively in C. You can transpose it back again when the variable is used, if required, but you might not want to (e.g., when printing a score in concert pitch, converting a trombone part from treble to bass clef, etc.) Mistakes in transpositions are less likely if all the music in variables is at a consistent pitch.

Also, only ever transpose to/from C. That means that the only other keys you will use are the natural pitches of the instruments - bes for a B-flat trumpet, aes for an A-flat clarinet, etc.

5.3 Large projects

When working on a large project, having a clear structure to your lilypond input files becomes vital.

- **Use a variable for each voice**, with a minimum of structure inside the definition. The structure of the `\score` section is the most likely thing to change; the `violin` definition is extremely unlikely to change in a new version of LilyPond.

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

- **Separate tweaks from music definitions**. This point was made previously, but for large projects it is absolutely vital. We might need to change the definition of `fthenp`, but then we only need to do this once, and we can still avoid touching anything inside `violin`.

```
fthenp = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c' {
  g4\fthenp c'8. e16
}
```

5.4 Troubleshooting

Sooner or later, you will write a file that LilyPond cannot compile. The messages that LilyPond gives may help you find the error, but in many cases you need to do some investigation to determine the source of the problem.

The most powerful tools for this purpose are the single line comment (indicated by `%`) and the block comment (indicated by `%{...%}`). If you don't know where a problem is, start commenting out huge portions of your input file. After you comment out a section, try compiling the file again. If it works, then the problem must exist in the portion you just commented. If it doesn't work, then keep on commenting out material until you have something that works.

In an extreme case, you might end up with only

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

(in other words, a file without any music)

If that happens, don't give up. Uncomment a bit – say, the bass part – and see if it works. If it doesn't work, then comment out all of the bass music (but leave `\bass` in the `\score` uncommented).

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

```
%}
}
```

Now start slowly uncommenting more and more of the `bass` part until you find the problem line.

Another very useful debugging technique is constructing *Secció “Tiny examples” in Informació general*.

5.5 Make and Makefiles

Pretty well all the platforms Lilypond can run on support a software facility called `make`. This software reads a special file called a **Makefile** that defines what files depend on what others and what commands you need to give the operating system to produce one file from another. For example the makefile would spell out how to produce ‘`ballad.pdf`’ and ‘`ballad.midi`’ from ‘`ballad.ly`’ by running Lilypond.

There are times when it is a good idea to create a **Makefile** for your project, either for your own convenience or as a courtesy to others who might have access to your source files. This is true for very large projects with many included files and different output options (e.g. full score, parts, conductor’s score, piano reduction, etc.), or for projects that require difficult commands to build them (such as `lilypond-book` projects). Makefiles vary greatly in complexity and flexibility, according to the needs and skills of the authors. The program GNU Make comes installed on GNU/Linux distributions and on MacOS X, and it is also available for Windows.

See the **GNU Make Manual** for full details on using `make`, as what follows here gives only a glimpse of what it can do.

The commands to define rules in a makefile differ according to platform; for instance the various forms of GNU/Linux and MacOS use `bash`, while Windows uses `cmd`. Note that on MacOS X, you need to configure the system to use the command-line interpreter. Here are some example makefiles, with versions for both GNU/Linux/MacOS and Windows.

The first example is for an orchestral work in four movements with a directory structure as follows:

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

```
| |-- symphonyI.ly
| |-- symphonyII.ly
| |-- symphonyIII.ly
| |-- symphonyIV.ly
|-- symphonyDefs.ily
```

The `.ly` files in the `Scores` and `Parts` directories get their notes from `.ily` files in the `Notes` directory:

```
%%% top of file "symphony-cello.ly"
\include ../symphonyDefs.ily
\include ../Notes/cello.ily
```

The makefile will have targets of `score` (entire piece in full score), `movements` (individual movements in full score), and `parts` (individual parts for performers). There is also a target `archive` that will create a tarball of the source files, suitable for sharing via web or email. Here is the makefile for GNU/Linux or MacOS X. It should be saved with the name `Makefile` in the top directory of the project:

Nota: When a target or pattern rule is defined, the subsequent lines must begin with tabs, not spaces.

```
# the name stem of the output files
piece = symphony
# determine how many processors are present
CPU_CORES=`cat /proc/cpuinfo | grep -m1 "cpu cores" | sed s/".*: "//`
# The command to run lilypond
LILY_CMD = lilypond -ddelete-intermediate-files \
            -dno-point-and-click -djob-count=$(CPU_CORES)

# The suffixes used in this Makefile.
.SUFFIXES: .ly .ily .pdf .midi

# Input and output files are searched in the directories listed in
# the VPATH variable. All of them are subdirectories of the current
# directory (given by the GNU make variable `CURDIR').
VPATH = \
    $(CURDIR)/Scores \
    $(CURDIR)/PDF \
    $(CURDIR)/Parts \
    $(CURDIR)/Notes

# The pattern rule to create PDF and MIDI files from a LY input file.
# The .pdf output files are put into the `PDF' subdirectory, and the
# .midi files go into the `MIDI' subdirectory.
%.pdf %.midi: %.ly
    $(LILY_CMD) $<; \
    if test -f "$*.pdf"; then \
        mv "$*.pdf" PDF/; \
    fi; \
    if test -f "$*.midi"; then \
        mv "$*.midi" MIDI/; \
    fi
```

```

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

# The dependencies of the movements.
$(piece)I.pdf: $(piece)I.ly $(notes)
$(piece)II.pdf: $(piece)II.ly $(notes)
$(piece)III.pdf: $(piece)III.ly $(notes)
$(piece)IV.pdf: $(piece)IV.ly $(notes)

# The dependencies of the full score.
$(piece).pdf: $(piece).ly $(notes)

# The dependencies of the parts.
$(piece)-cello.pdf: $(piece)-cello.ly cello.ily
$(piece)-horn.pdf: $(piece)-horn.ly horn.ily
$(piece)-oboes.pdf: $(piece)-oboes.ly oboe.ily
$(piece)-viola.pdf: $(piece)-viola.ly viola.ily
$(piece)-violinOne.pdf: $(piece)-violinOne.ly violinOne.ily
$(piece)-violinTwo.pdf: $(piece)-violinTwo.ly violinTwo.ily

# Type `make score' to generate the full score of all four
# movements as one file.
.PHONY: score
score: $(piece).pdf

# Type `make parts' to generate all parts.
# Type `make foo.pdf' to generate the part for instrument `foo'.
# Example: `make symphony-cello.pdf'.
.PHONY: parts
parts: $(piece)-cello.pdf \
    $(piece)-violinOne.pdf \
    $(piece)-violinTwo.pdf \
    $(piece)-viola.pdf \
    $(piece)-oboes.pdf \
    $(piece)-horn.pdf

# Type `make movements' to generate files for the
# four movements separately.
.PHONY: movements
movements: $(piece)I.pdf \
    $(piece)II.pdf \
    $(piece)III.pdf \
    $(piece)IV.pdf

all: score parts movements

archive:

```

```
tar -cvvf stamitz.tar \      # this line begins with a tab
--exclude=*pdf --exclude=*~ \
--exclude=*midi --exclude=*.tar \
../Stamitz/*
```

There are special complications on the Windows platform. After downloading and installing GNU Make for Windows, you must set the correct path in the system's environment variables so that the DOS shell can find the Make program. To do this, right-click on "My Computer," then choose **Properties** and **Advanced**. Click **Environment Variables**, and then in the **System Variables** pane, highlight **Path**, click **edit**, and add the path to the GNU Make executable file, which will look something like this:

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

The makefile itself has to be altered to handle different shell commands and to deal with spaces that are present in some default system directories. The **archive** target is eliminated since Windows does not have the **tar** command, and Windows also has a different default extension for midi files.

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

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

.SUFFIXES: .ly .ily .pdf .mid

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

%.pdf %.mid: %.ly
    $(LILY_CMD) $<      # this line begins with a tab
    if exist "$*.pdf" move /Y "$*.pdf" PDF/ # begin with tab
    if exist "$*.mid" move /Y "$*.mid" MIDI/ # begin with tab

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

$(piece)I.pdf: $(piece)I.ly $(notes)
$(piece)II.pdf: $(piece)II.ly $(notes)
```

```

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

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

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

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

.PHONY: parts
parts: $(piece)-cello.pdf \
       $(piece)-violinOne.pdf \
       $(piece)-violinTwo.pdf \
       $(piece)-viola.pdf \
       $(piece)-oboes.pdf \
       $(piece)-horn.pdf

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

all: score parts movements

```

The next Makefile is for a `lilypond-book` document done in LaTeX. This project has an index, which requires that the `latex` command be run twice to update links. Output files are all stored in the `out` directory for `.pdf` output and in the `htmlout` directory for the `html` output.

```

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

all: pdf web keep

pdf:
    $(LILYBOOK_PDF) # begin with tab
    $(PDF)          # begin with tab

```

```

$(INDEX)          # begin with tab
$(PDF)            # begin with tab
$(PREVIEW)        # begin with tab

web:
$(LILYBOOK_HTML) # begin with tab
$(HTML)          # begin with tab
cp -R $(WEBDIR)/$(FILE)/ ./ # begin with tab
$(BROWSER) $(FILE)/$(FILE).html & # begin with tab

keep: pdf
cp $(OUTDIR)/$(FILE).pdf $(FILE).pdf # begin with tab

clean:
rm -rf $(OUTDIR) # begin with tab

web-clean:
rm -rf $(WEBDIR) # begin with tab

archive:
tar -cvvf myproject.tar \ # begin this line with tab
--exclude=out/* \
--exclude=htmlout/* \
--exclude=myproject/* \
--exclude=*midi \
--exclude=*pdf \
--exclude=*~ \
../MyProject/*

```

TODO: make this thing work on Windows

The previous makefile does not work on Windows. An alternative for Windows users would be to create a simple batch file containing the build commands. This will not keep track of dependencies the way a makefile does, but it at least reduces the build process to a single command. Save the following code as `build.bat` or `build.cmd`. The batch file can be run at the DOS prompt or by simply double-clicking its icon.

```

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

```

Vegeu també

This manual: [\[Command-line usage\]](#), pàgina [\[undefined\]](#), Capítol 3 [\[lilypond-book\]](#), pàgina 20

Annex A GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Annex B Índex del LilyPond

\		
\header in L ^A T _E X documents	25	
A		
ABC	44	
actualització d'un fitxer del LilyPond	16	
actualització de fitxers d'entrada antics	16	
advertiment	11	
apuntar i clicar, línia d'ordres	4	
C		
carpeta, dirigir la sortida cap a	4	
cerca, ruta de	2	
chroot, executar dins d'una gàbia	2	
Coda Technology	45	
coloring, syntax	41	
convert-ly	16	
crides, traça de	11	
D		
docbook	20	
DocBook, adding music	20	
documents, adding music	20	
dvips	31	
E		
editors	41	
emacs	41	
enigma	45	
error	11	
error de programació	11	
error del Scheme	11	
error fatal	11	
error, format dels missatges de	12	
error, missatges d'error	11	
ETF	45	
Evince	40	
expressions del Scheme, avaluació	2	
External programs, generating LilyPond files	45	
F		
fatal, error	11	
file size, output	40	
Finale	45	
fitxers, cerca de	2	
format, sortida	2	
H		
HTML	20	
HTML, adding music	20	
I		
invocació de lilypond	2	
invoking dvips	31	
L		
LANG	9	
LaTeX	20	
LaTeX, adding music	20	
LibreOffice.org	46	
LILYPOND_DATADIR	9	
línia d'ordres, opcions de	2	
loglevel	3	
M		
make	51	
makefiles	51	
Manuals	1	
MIDI	42	
missatges d'error	11	
modes, editor	41	
modificadors	2	
musicology	20	
MusicXML	43	
O		
opcions de la línia d'ordres per a lilypond	2	
OpenOffice.org	46	
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	
R		
registre, nivell de	3	
S		
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	
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

type1 fonts	31
-------------------	----

V

vim	41
-----------	----

X

Xpdf	39
------------	----