LilyPond

Das Notensatzprogramm

Usage

Das LilyPond-Entwicklerteam

Diese Datei erklärt, wie man die Programme, die mit LilyPond Version 2.13.39 verteilt werden, benutzt werden. Zusätzlich werden einige Hinweise zur effizienten Benutzung der Programme vorgestellt.

Zu mehr Information, wie dieses Handbuch unter den anderen Handbüchern positioniert, oder um dieses Handbuch in einem anderen Format zu lesen, besuchen Sie bitte Abschnitt “Manuals” in *Allgemeine Information*.

Wenn Ihnen Handbücher fehlen, finden Sie die gesamte Dokumentation unter `http://www.lilypond.org/`.

Copyright © 1999–2010 bei den Autoren.

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

Die Übersetzung der folgenden Lizenzanmerkung ist zur Orientierung für Leser, die nicht Englisch sprechen. Im rechtlichen Sinne ist aber nur die englische Version gültig.

> Es ist erlaubt, dieses Dokument unter den Bedingungen der GNU Free Documentation Lizenz (Version 1.1 oder spätere, von der Free Software Foundation publizierte Versionen, ohne invariante Abschnitte), zu kopieren, zu verbreiten und/oder zu verändern. Eine Kopie der Lizenz ist im Abschnitt “GNU Free Documentation License” angefügt.
>
> 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”.

Für LilyPond Version 2.13.39

Inhaltsverzeichnis

1 `lilypond` starten ... 1

1.1 Übliche Programmbenutzung ... 1
1.2 Benutzung auf der Kommandozeile ... 1
1.2.1 `lilypond` aufrufen ... 1
Häufige Kommandozeilenbefehle ... 2
1.2.2 Optionen auf der Kommandozeile für `lilypond` ... 2
1.2.3 Umgebungsvariablen ... 6
LilyPond in chroot-Kerker ... 6
1.3 Fehlermeldungen ... 8
1.4 Häufige Fehlermeldungen ... 9
Noten laufen aus der Seite heraus ... 9
Ein zusätzliches System erscheint ... 9
Offensichtlicher Fehler in ../ly/init.ly ... 10
Fehlermeldung Ungebundene Variable % ... 11
Fehlermeldung FT_Get_Glyph_Name ... 11
Warnung über absteigende staff affinities ... 11

2 Dateien mit `convert-ly` aktualisieren ... 12

2.1 Warum verändert sich die Syntax? ... 12
2.2 `convert-ly` aufrufen ... 12
2.3 Optionen auf der Kommandozeile für `convert-ly` ... 13
2.4 Probleme mit `convert-ly` ... 13
2.5 Manuelle Konversion ... 14

3 `lilypond-book` aufrufen ... 16

3.1 Ein musikwissenschaftlicher Text als Beispiel ... 16
3.2 Noten in Text integrieren ... 19
3.2.1 LaTeX ... 19
3.2.2 Texinfo ... 21
3.2.3 HTML ... 21
3.2.4 DocBook ... 22
3.3 Die Musikfragment-Optionen ... 23
3.4 `lilypond-book` aufrufen ... 26
3.5 Dateiendungen ... 28
3.6 `lilypond-book`-Vorlagen ... 29
3.6.1 LaTeX ... 29
3.6.2 Texinfo ... 29
3.6.3 html ... 30
3.6.4 xelatex ... 30
3.7 Das Inhaltsverzeichnis flexibel einsetzen ... 31
3.8 Alternative Methoden Text und Musik zu kombinieren ... 33

4 Externe Programme ... 34

4.1 Point and click ... 34
4.2 Unterstützung von Texteditoren ... 35
Emacs-Modus ... 35
Vim-Modus ... 35

Andere Editoren ... 35
4.3 Von anderen Formaten konvertieren ... 35
4.3.1 `midi2ly` aufrufen ... 36
4.3.2 `musicxml2ly` aufrufen ... 37
4.3.3 `abc2ly` aufrufen ... 38
4.3.4 `etf2ly` aufrufen ... 39
4.3.5 Andere Formate ... 39
4.4 LilyPond-Ausgabe in anderen Programmen ... 39
4.4.1 Viele Zitate aus einer langen Partitur ... 39
4.4.2 LilyPond-Noten in OpenOffice.org integrieren ... 40
4.4.3 LilyPond-Noten in andere Programme integrieren ... 40
4.5 Unabhängige `include`-Abschnitte ... 40
4.5.1 MIDI-Artikulation ... 40

5 Vorschläge zum Schreiben von LilyPond-Eingabe-Dateien ... 41
5.1 Allgemeine Vorschläge ... 41
5.2 Das Kopieren von bereits vorhandener Musik ... 42
5.3 Große Projekte ... 42
5.4 Fehlersuche ... 43
5.5 Make und Makefiles ... 43

Anhang A GNU Free Documentation License ... 50

Anhang B LilyPond-Index ... 57

1 lilypond starten

Dieses Kapitel behandelt die technischen Details, wie Lilypond ausgeführt werden kann.

1.1 Übliche Programmbenutzung

Die meisten Benutzer führen LilyPond von einer graphischen Benutzeroberfläche aus. Siehe Abschnitt “Übung” in *Handbuch zum Lernen*, falls Sie dies nicht bereits getan haben. Wenn Sie einen alternativen Editor für Ihre LilyPond-Dateien verwenden, lesen Sie bitte die Dokumentation dieses Editors.

1.2 Benutzung auf der Kommandozeile

Dieser Abschnitt enthält zusätzliche Informationen, wie Sie LilyPond von der Kommandozeile ausführen können. Dies kann erforderlich sein, um etwa zusätzliche Optionen an das Programm zu übergeben. Außerdem sind einige Zusatzprogramme (wie etwa `midi2ly`) nur von der Kommandozeile verfügbar.

Unter ‚Kommandozeile‘ verstehen wir die Kommandozeile des jeweiligen Betriebssystems. Windows Benutzern ist sie vielleicht eher unter den englischen Begriffen ‚DOS shell‘ oder ‚command shell‘ bekannt. MacOS X Benutzer kennen sie eher unter ‚Terminal‘ oder ‚Konsole‘. Einige zusätzliche Einrichtungsarbeiten werden unter MacOS X, siehe Abschnitt “MacOS X” in *Allgemeine Information*.

Wie die Kommandozeile im jeweiligen Betriebssystem benutzt werden kann, soll in diesem Handbuch nicht näher beschrieben werden. Sehen Sie bitte im Handbuch Ihres Betriebssystems nach oder informieren Sie sich im Internet, wenn Sie mit der Kommandozeile nicht vertraut sind.

1.2.1 lilypond aufrufen

Das `lilypond` Programm kann folgendermaßen von der Kommandozeile aufgerufen werden.

```
lilypond [Option]... Dateiname...
```

Wird ein ‘`Dateiname`’ ohne Erweiterung angegeben, so wird ‘`.ly`’ als Standarderweiterung für LilyPond-Dateien benutzt. Um Daten von `stdin` einzulesen, benutzen Sie einfach einen Bindestrich (`-`) als *Dateiname*.

Wenn Lilypond die Datei ‘`Dateiname.ly`’ verarbeitet, werden daraus die Dateien ‘`Dateiname.ps`’ und ‘`Dateiname.pdf`’ erzeugt. Es können an `lilypond` auch mehrere ‘`.ly`’ Dateien übergeben werden, die dann einzeln und voneinander unabhängig abgearbeitet werden.[1]

Falls ‘`Dateiname.ly`’ mehr als eine `\book`-Umgebung enthält, werden die weiteren Stücke in durchnummerierte Dateien der Form ‘`Dateiname-1.pdf`’ ausgegeben. Zusätzlich wird der Wert der Variable `output-suffix` zwischen den ursprünglichen Dateinamen und der Zahl eingefügt. Eine Lilypond-Datei *Dateiname.ly* mit dem Inhalt

```
#(define output-suffix "Geige")
\score { ... }
#(define output-suffix "Cello")
\score { ... }
```

erzeugt daher die Dateien *Dateiname*‘`-Geige.pdf`’ und *Dateiname*‘`-Cello-1.pdf`’.

[1] Der Zustand von GUILE wird allerdings nicht nach jeder Datei zurückgesetzt, sodass Achtung geboten ist, wenn in einer Datei globale Änderungen von Scheme aus durchgeführt werden.

Häufige Kommandozeilenbefehle

Wenn Ihre Kommandozeile normale Weiterleitungen unterstützt, können Sie es nützlich finden, mit folgenden Befehlen die Ausgabe der Kommandozeile in eine Datei zu leiten:

- `lilypond file.ly 1>stdout.log` um normale Ausgabe zu erhalten
- `lilypond file.ly 2>stderr.log` um Fehlermeldungen zu erhalten
- `lilypond file.ly &>all.log` um alle Meldungen zu erhalten

Wenden Sie sich an die Dokumentation für Ihre Kommandozeile, um zu sehen, ob derartige Optionen unterstützt werden oder die Syntax unterschiedlich ist. Beachten Sie, dass es sich hier um reine Verwaltungsprogramme handelt, die nichts mit LilyPond zu tun haben.

1.2.2 Optionen auf der Kommandozeile für `lilypond`

Die folgenden Kommandozeilenoptionen werden von `lilypond` unterstützt:

`-e,--evaluate=`*expr*

Wertet den Scheme-Ausdruck *expr* aus, bevor die '`.ly`' Dateien gelesen und interpretiert werden. Die `-e` Option kann auch mehrfach angegeben werden, die Ausdrücke werden nacheinander ausgewertet.

Da der Ausdruck im `guile-user` Modul ausgewertet wird, ist bei der Definitionen innerhalb von *expr* folgendes Vorgehen nötig. An der Kommandozeile wird z.B. `a` im `guile-user` Modul definiert:

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

Am Beginn der '`.ly`'-Datei muss dann das `guile-user` Modul noch geladen werden, bevor die Definition von *a* verfügbar ist:

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

`-f,--format=`*Format*

Bestimmt das Ausgabeformat. Mögliche Werte von *Format* sind `svg`, `ps`, `pdf` und `png`.

Beispiel: `lilypond -fpng` *Dateiname*`.ly`

`-d,--define-default=`*Variable*`=`*Wert*

Damit wird die interne Programmoption *Variable* auf den Scheme-Wert *Wert* gesetzt. Wird kein *Wert* angegeben, so wird *#t* benutzt. Um eine Option auszuschalten, kann der Präfix `no-` dem Namen *Variable* der Variable vorangestellt werden. So ist etwa

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

dasselbe wie

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

Hier sind ein paar interessante Optionen:

'`help`' Die Ausführung von `lilypond -dhelp` zeigt alle verfügbaren `-d` Optionen.

'`paper-size`'

Setzt das Standard-Papierformat,

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

Die Zeichenkette, die das Format angibt, muss in Anführungszeichen mit Backslash (`\"`) stehen.

'`safe`' Vertraut der `.ly` Datei nicht.

Wenn LilyPond über einen Webserver verfügbar gemacht wird, **MUSS** unbedingt eine die Optionen `--safe` oder `--jail` angegeben werden. Die `--safe` Option verhindert, dass in der .ly-Datei angegebener Scheme-Code das System gefährden kann, wie etwa in folgendem Beispiel:

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

Mit der `-dsafe` Option werden alle Scheme-Ausdrücke einem speziellen sicheren Modus ausgewertet. Dieser Modus ist vom GUILE '`safe-r5rs`' Modul abgeleitet und fügt noch zahlreiche weitere erlaubte Funktionen der LilyPond Programm-Schnittstelle hinzu. Diese Funktionen sind in '`scm/safe-lily.scm`' angegeben.

Zusätzliche verbietet der sichere Modus auch `\include` Befehle.

Im sicheren Modus ist es nicht möglich, LilyPond-Variablen nach Scheme zu exportieren.

`-dsafe` erkennt jedoch *KEINE* Überbeanspruchung der verfügbaren Ressourcen. In diesem Modus ist es also trotzdem möglich, dass LilyPond in einer Endlosschleife hängt, z.B. wenn zyklische Datenstrukturen an das Backend übergeben werden. Wenn LilyPond also auf einem öffentlich zugänglichen Webserver verfügbar gemacht wird, sollte der Prozess sowohl in der CPU- als auch in der Speichernutzung limitiert werden.

Der sichere Modus verhindert auch, dass zahlreiche nützliche Musikfragmente von LilyPond verarbeitet werden. Die `--jail` Option ist eine sicherere Alternative, benötigt allerdings auch mehr Aufwand zur Einrichtung.

'`backend`' Gibt an, welches Ausgabeformat das LilyPond Backend benutzt. Mögliche Werte für diese Option sind:

`ps` PostScript-Ausgabeformat.

Postscript-Dateien enthalten auch TTF-, Type1- und OTF-Schriften. Allerdings wird die gesamte Schriftart eingefügt und nicht nur die benötigten Zeichen. Vor allem wenn nicht-westliche Zeichensätze benutzt werden, kann dies zu sehr großen Dateien führen.

`eps` Erzeugt ‚encapsulated PostScript' (EPS). Jede Seite (oder jedes System) wird als eigene '`EPS`'-Datei ausgegeben, inklusive Schriftarten. Außerdem wird eine Datei mit allen Seiten (bzw. Systemen) und Schriftarten erzeugt.

Dies ist die Standardeinstellung von `lilypond-book`.

`svg`

SVG-Ausgabe (Scalable Vector Graphics).

Hiermit wird eine einzelne SVG-Datei ohne eingebundene Schriften für jede Seite der Partitur erstellt. Es wird empfohlen, Century Schoolbook-Schriftarten zu installieren, die auch in der LilyPond-Installation enthalten sind, um optimales Rendern zu erhalten.

Unter UNIX können diese Schriftarten einfach aus dem LilyPond-Verzeichnis (normalerweise '`/usr/share/lilypond/VERSION/fonts/otf/`') nach '`~/.fonts`' kopiert werden. Die SVG-Ausgabe sollte mit allen SVG-Editoren oder Betrachtungsprogrammen kompatibel sein.

`scm` gibt die rohen Scheme-basierenden Zeichenbefehle aus, wie sie intern von LilyPond benutzt werden.

`null` Keine Partitur wird ausgegeben, hat gleichen Effekt wie `-dno-print-pages`.

Beispiel: `lilypond -dbackend=svg` *`Dateiname`*`.ly`

'`preview`' Erzeugt eine Ausgabedatei, die nur die Titelzeilen und das erste System enthält. Wenn `\bookpart`-Umgebungen benutzt werden, erscheinen die Titel und ersten Systeme jedes `\bookpart`-Abschnitts in der Datei. Die Backends `ps`, `eps` und `svg` unterstützen diese Option.

'`print-pages`'
Erzeugt vollständige Seiten (Standardeinstellung). `-dno-print-pages` ist in Verbindung mit `-dpreview` nützlich.

`-h,--help`
Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-H,--header=`*`FELD`*
Gibt den Inhalt eines Feldes aus dem `\header`-Block in die Datei '`Dateiname.`*`FELD`*' aus.

`--include, -I=`*`Verzeichnis`*
Fügt *Verzeichnis* zur Liste der Suchpfade hinzu.

Mehrere -I-Optionen können angegeben werden. Die Suche beginnt mit dem ersten definierten Verzeichnis und setzt in den weiteren Verzeichnissen fort, wenn die gesuchte Datei nicht in dem Verzeichnis gefunden wird.

`-i,--init=`*`Initialisierungsdatei`*
Benutzt *Initialisierungsdatei* zur gesamten Programminitialisierung. Der Standardwert ist '`init.ly`'.

`-o,--output=`*`DATEI`*
Schreibt das Ergebnis der Verarbeitung mit LilyPond in die Ausgabedatei *DATEI*. Wenn ein Verzeichnis mit dem Namen existiert, werden die Ausgabedateien in dieses Verzeichnis gespeichert, wobei der Dateiname der Eingabedatei benutzt wird. Die entsprechende Dateinamenserweiterung wird angehängt (z.B. `.pdf` für pdf).

`--ps` Erzeugt PostScript.

`--png` Erzeugt eine Grafik-Datei im PNG-Format von jeder Seite. Diese Option impliziert auch `--ps`. Die Auflösung in DPI der Grafik kann festgelegt werden durch

```
-dresolution=110
```

`--pdf` Erzeugt PDF-Dateien. Dies impliziert `--ps`.

`-j,--jail=`*`Benutzer,Gruppe,Jail-Verzeichnis,Arbeitsverzeichnis`*
Führt `lilypond` in einem chroot-Jail aus.

Die `--jail` Option ist eine flexiblere Alternative zu `--safe`, wenn LilyPond über das Internet verfügbar gemacht wird oder LilyPond-Quelldateien von Dritten automatisch verarbeitet werden.

Sie funktioniert dergestalt, dass das Wurzelverzeichnis von `lilypond` auf *Jail-Verzeichnis* gesetzt wird, bevor die tatsächliche Kompilierung der .ly-Datei beginnt. Der Benutzer und die Gruppe werden auf die angegebenen Werte gesetzt und das aktuelle Arbeitsverzeichnis wird ebenfalls auf den angegebenen Wert *Arbeitsverzeichnis* gesetzt. Diese Einstellungen garantieren (zumindest in der Theorie), dass es nicht möglich ist, aus dem Jail auszubrechen. Damit `--jail` funktioniert, muss `lilypond` allerdings als root ausgeführt werden, was normalerweise auf sichere Art mit dem Kommando `sudo` erreicht werden kann.

Das Jail-Verzeichnis zu erstellen ist etwas heikel, da LilyPond alle zur Ausführung nötigen Bibliotheken und Dateien *innerhalb des Jail-Verzeichnisses* finden muss. Ein typisches Setup besteht aus folgenden Punkten:

Erstellung eines getrennten Dateisystems
: Ein eigenes Dateisystem muss für LilyPond erstellt werden, sodass es mit sicheren Einstellungen wie `noexec`, `nodev` und `nosuid` eingebunden werden kann. Damit ist es unmöglich, Programme von diesem Dateisystem auszuführen oder direkt auf eine Hardware-Schnittstelle zuzugreifen. Wenn Sie keine eigene Partition erstellen möchten, können Sie auch eine Datei der entsprechenden Größe erstellen und sie als ‚loop'-Gerät einbinden. Ein getrenntes Dateisystem garantiert auch, dass LilyPond nicht mehr Festplattenspeicher benutzt als erlaubt.

Erstellung eines eigenen Benutzerkontos
: Es sollte ein eigener Benutzer und eine eigene Gruppe (z. B. `lily/lily`) mit geringen Rechten für die Ausführung von LilyPond innerhalb des Jails benutzt werden. Nur ein einziges Verzeichnis des Jails sollte für den Benutzer schreibbar sein und als *Arbeitsverzeichnis* an `lilypond` übergeben werden.

Einrichtung des Jails
: LilyPond muss zahlreiche Dateien für die Ausführung einlesen. All diese Dateien müssen in das Jail-Verzeichnis kopiert werden (mit denselben Pfaden wie im tatsächlichen Wurzel-Dateisystem). Die gesamte LilyPond-Installation (typischerweise '`/usr/share/lilypond`') sollte kopiert werden.

  Falls Probleme auftreten, ist es am einfachsten, Lilypond mittels `strace` zu starten, wodurch Sie relativ leicht feststellen können, welche Dateien im Jail noch fehlen.

Ausführung von LilyPond
: In einem mit `noexec` eingebundenen Jail ist es nicht möglich, externe Programme auszuführen. Daher muss LilyPond auf eine Art gestartet werden, die keine weitere Ausführung von Programmen benötigt. Wie bereits erwähnt muss LilyPond mit Administrator-Rechten gestartet werden (die es allerdings sofort wieder abgibt), beispielsweise mittels `sudo`. Außerdem ist es eine gute Idee, die LilyPond zur Verfügung stehende CPU-Zeit zu limitieren (z. B. mit `ulimit -t`) und – falls das Betriebssystem dies unterstützt – auch den zur Verfügung stehenden Hauptspeicher.

`-v,--version`
: Gibt die Versionsnummer aus.

`-V,--verbose`
: Gibt ausführliche informative Meldungen aus: Zeigt die vollen Dateipfade aller gelesenen Dateien sowie Informationen über die Laufzeit.

`-w,--warranty`
: Zeigt die Garantiebedingungen an, unter denen GNU LilyPond steht. (Es besteht **KEINERLEI GARANTIE**!)

1.2.3 Umgebungsvariablen

`lilypond` erkennt und benützt die folgenden Umgebungsvariablen:

`LILYPOND_DATADIR`
: Diese Variable gibt das Verzeichnis an, wo Lilypond seine eigenen Dateien, Meldungen und Übersetzungen finden kann. Dieses Verzeichnis sollte Unterverzeichnisse '`ly/`', '`ps/`', '`tex/`', etc. beinhalten.

`LANG`
: Gibt die Sprache an, in der Warnungen und Fehlermeldungen ausgegeben werden.

`LILYPOND_GC_YIELD`
: Mit dieser Variable (mit Werten zwischen 0 und 100) kann die Feinabstimmung zwischen dem Bedarf an Hauptspeicher und Rechenleistung bei der Ausführung von LilyPond durchgeführt werden. Bei höheren Werten benutzt LilyPond mehr Hauptspeicher, benötigt aber weniger Prozessor-Leistung. Bei niedrigeren Werten wird mehr Prozessor-Zeit, dafür weniger Hauptspeicher benötigt. Voreinstellung ist ein Wert von `70`.

LilyPond in chroot-Kerker

Einen Server einzurichten, der LilyPond in einem chroot-Kerker bedient, ist recht kompliziert. Die einzelnen Schritten finden sich unten aufgeliestet. Beispiele sind für Ubuntu Linux und erfordern evtl. die Benutzung von `sudo` an den entsprechenden Stellen.

- Installieren Sie die nötigen Pakete: LilyPond, GhostScript und ImageMagick.
- Erstellen Sie einen neuen Benutzer mit dem Namen `lily`:

  ```
  adduser lily
  ```

  Hierdurch wird auch eine Gruppe `lily` und ein Heimat-Ordner `/home/lily` für den neuen Benutzer erstellt.
- Im Heimat-Ordner des Benutzers `lily` erstellen Sie eine Datei, die als eigenes Dateisystem eingesetzt wird:

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

  In diesem Beispiel wird eine 200-MB-Datei als Kerker-Dateisystem erstellt.
- Erstellen Sie ein loop device, erstellen Sie ein Dateisystem und mounten Sie es, dann erstellen Sie dort einen Ordner, in dem der Benutzer `lily` Schreibrechte hat:

  ```
  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
  ```
- In der Konfiguration des Servers ist der Kerker (JAIL) `/mnt/lilyloop` und das Verzeichnis (DIR) `/lilyhome`.
- Erstellen Sie einen großen Verzeichnisbaum in dem Kerker, indem Sie die notwendigen Dateien dorthin kopiert, wie das Beispielskript unten zeigt.

Sie könne sed benutzen, um die notwendigen Kopierbefehle für ein bestimmtes Programm zu erstellen:

```
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
```

Beispiel-Skript für 32-bit Ubuntu 8.04

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

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

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

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

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

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

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

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

1.3 Fehlermeldungen

Während der Verarbeitung einer Dateien können diverse Meldungen an der Kommandozeile auftreten:

Warnung (Warning)
: Irgendetwas ist verdächtig. Wenn Sie etwas Ungewöhnliches in Ihrer Datei durchführen, dann werden Sie die Meldung verstehen und können sie gegebenenfalls ignorieren. Im Normalfall jedoch bedeutet eine Warnung, dass mit Ihrer Datei etwas nicht stimmt, LilyPond jedoch trotzdem versucht, die Datei soweit wie möglich korrekt zu übersetzen.

Fehler (Error)
: Irgendetwas stimmt definitiv nicht. Der aktuelle Bearbeitungsschritt (Einlesen, Interpretieren oder Formatieren der Datei) wird noch fertig ausgeführt, danach bricht die Bearbeitung aber ab.

Fataler Fehler (Fatal error)
: Irgendetwas stimmt definitiv nicht und LilyPond kann nicht weiter ausgeführt werden. Dies ist nur sehr selten der Fall, meist sind die Schriftarten nicht korrekt installiert.

Scheme Fehler (Scheme error)
: Fehler, die während der Ausführung von Scheme-Code auftreten, werden vom Scheme-Interpreter aufgefangen und an der Kommandozeile ausgegeben. Wenn Sie LilyPond mit der `--verbose` Option (auch `-V`) ausführen, wird der sogenannte ‚Call trace' ausgegeben, der die aufgerufenen Funktionen zur Zeit des Fehlers angibt.

Programmierfehler (Programming error)
: Eine interne Inkonsistenz ist aufgetreten. Diese Fehlermeldungen sollen den Programmierern die Fehlersuche erleichtern und können meistens einfach ignoriert werden. In manchen Fällen werden so viele Meldungen ausgegeben, dass die Lesbarkeit der restliche Ausgabe davon beeinträchtigt wird.

Abgebrochen (core dumped)
: Dies bezeichnet einen ernsten Programmierfehler, der das Programm zum Absturz gebracht hat. Solche Fehler werden als kritisch angesehen. Falls daher einer auftritt, senden Sie bitte einen Bug-Report!

Wenn Warnungen oder Fehlermeldungen mit einer konkreten Stelle in der Eingabedatei verknüpft werden können, dann hat die Meldung die folgende Form:

```
Dateiname:Zeile:Spalte: Meldung
Fehlerhafte Eingabezeile
```

Ein Zeilenumbruch wird in der fehlerhaften Zeile an jener Stelle eingefügt, wo der Fehler aufgetreten ist. Zum Beispiel

```
test.ly:2:19: Fehler: keine gültige Dauer: 5
  { c'4 e'
            5 g' }
```

Diese Stellen sind LilyPonds Vermutung, wo die Warnung oder der Fehler aufgetreten ist, allerdings treten Warnungen und Fehler ja gerade in unerwarteten Fällen auf. Manchmal kann Lilypond auch eine fehlerhafte Stelle zwar noch problemlos verarbeiten, ein paar Zeilen später wirkt sich der Fehler aber dann doch noch aus. In solchen Fällen, wo Sie in der angegebenen Zeile keinen Fehler erkennen, sollten Sie auch die Zeilen oberhalb der angegebenen Stelle genauer unter die Lupe nehmen.

Mehr Information darüber findet sich in Abschnitt 1.4 [Häufige Fehlermeldungen], Seite 9.

1.4 Häufige Fehlermeldungen

Die Fehlermeldungen, die unten beschrieben werden, treten of auf, es ist jedoch nicht immer einfach, die Urache zu finden. Wenn der Fehler einmal verstanden ist, kann er einfach behoben werden.

Noten laufen aus der Seite heraus

Noten, die rechts aus der Seite herauslaufen oder sehr komprimiert aussehen, liegen in fast allen Fällen an einer falschen Notendauer einer Note, die dazu fürt, dass die letzte Note im Takt über die Taktgrenze hinwegdauert. Es ist nicht falsch, wenn die letzte Note eines Taktes über den Takt hinausdauert, weil einfach angenommen wird, dass sie im nächsten Takt fortgesetzt wird. Aber wenn eine längere Sequenz dieser überhängenden Noten auftritt, können die Noten sehr gedrängt aussehen oder über den Seitenrand fließen, weil die automatische Zeilenumbruchsfunktion einen Umbruch nur am Ende eines vollständigen Taktes einfügen kann, also wenn alle Noten zum Ende des Taktstriches auch wirklich aufhören.

> **Achtung:** Eine falsche Dauer kann dazu führen, dass Zeilenumbrüche nicht möglich sein und die Zeile entweder sehr gedrängt dargestllt wird oder über den Seitenrand fließt.
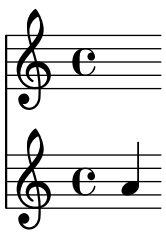
Die falsche Dauer kann einfach gefunden werden, wenn Taktstrichüberprüfung eingesetzt wird, siehe Abschnitt “Takt- und Taktzahlüberprüfung” in *Notationsreferenz*.

If you actually intend to have a series of such carry-over measures you will need to insert an invisible bar line where you want the line to break. For details, see Abschnitt “Taktstriche” in *Notationsreferenz*.

Ein zusätzliches System erscheint

Wenn Kontext nicht explizit mit `\new` oder `\context` erstellt werden, werden sie vom Programm erstellt, sobald ein Befehl angetroffen wird, der im aktuellen Kontext nicht funktioniert. In einfachen Partituren ist diese automatische Erstellung sehr nützlich und die meisten Beispiele der LilyPond-Handbücher benutzen diese Schreiberleicherterung. Manchmal jedoch kann es vorkommen, dass durch die automatische Erstellung von Systemen aufeinmal unerwartete Notensysteme erstellt werden. Beispielsweise könnte man annehmen, dass folgendes Beispiel alle Notenköpfe in dem Notensystem rot macht, aber als Resultat hat man zwei Systeme, während die Notenköpfe immernoch schwarz im unteren System erscheinen.

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



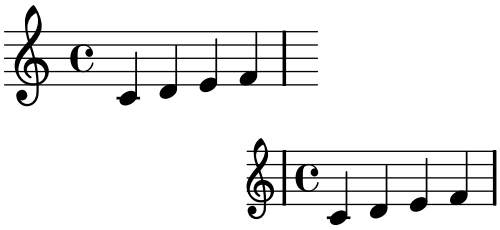
Das liegt daran, dass kein Staff-Kontext existiert, wenn der \override-Befehl verarbeitet wird, sodass ein System für diesen Befehl erstellt wird. Dann aber erstellt \new Staff noch ein zusätzliches System, wo die Noten gesetzt werden. Die richtige Schreibweise wäre:

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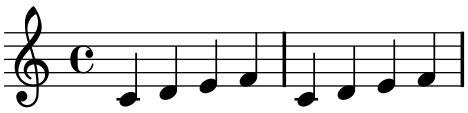


Ein zweites Beispiel zeigt, dass ein \relative-Befehl innerhalb von \repeat zwei Systeme erstellt, wobei der zweite etwas verschoben ist. Das liegt daran, dass \repeat zwei \relative-Umgebungen erstellt, die jede implizit einen Staff- und Voice-Kontext erstellen.

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



Indem man die Voice-Kontexte explizit erstellt, kann das Problem umgangen werden.

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



Offensichtlicher Fehler in ../ly/init.ly

Verschiedene seltsame Fehlermeldungen können über Syntax-Fehler in ../ly/init.ly auftauchen, wenn die Eingabedatei nicht richtig formuliert ist, wenn sie etwa nicht richtig passende Klammerpaare oder Anführungszeichen enthält.

Der üblichste Fehler ist das Fehlen einer geschweiften Klammer (}) am Ende der score-Umbgebung. Die Lösung ist hier klar: überprüfen Sie, ob die score-Umgebung richtig beendet wurde. Die richtige Struktur einer Eingabedatei wird beschrieben in Abschnitt “Wie eine LilyPond-Eingabe-Datei funktioniert” in *Handbuch zum Lernen*. Ein Editor, der die Klammerpaare automatisch anzeigt, ist sehr hilfreich, um derartige Fehler zu vermeiden.

Eine weitere übliche Fehlerquelle ist kein Leerzeichen zwischen der letzten Silbe einer lyrics-Umgebung und der schließenden Klammer (}). Ohne diese Trennung wird die Klammer als Teil der Silbe gewertet. Es bietet sich immer an, Leerzeichen vor und hinter *jede* Klammer zu setzen. Wie wichtig das ist, wenn Gesangstext eingesetzt wird, siehe Abschnitt “Was ist Gesangtext” in *Notationsreferenz*.

Diese Fehlermeldung kann auch mit einem fehlenden schließenden Anführungszeichen (`"`) auftreten. In diesem Fall sollte die begleitende Fehlermeldung eine Zeilenzahl angeben, die dicht am Fehler liegt. Die nicht paarigen Anführungszeichen sind meistens ein oder zwei Zeilen darüber.

Fehlermeldung Ungebundene Variable %

Diese Fehlermeldung erscheint am Ende der Kommandozeilenausgabe oder in der Log-Datei mit einer Meldung „GUILE signalled an error ...“ jedes Mal, wenn eine Scheme-Routine aufgerufen wird, die (falscherweise) ein *LilyPond*-Kommentar und kein *Scheme*-Kommentar enthält.

LilyPond-Kommentare befginnen mit dem Prozent-Zeichen (`%`) und dürfen nicht in Scheme-Routinen benutzt werden. Scheme-Kommentare beginnen mit einem Semikolon (`;`).

Fehlermeldung FT_Get_Glyph_Name

Diese Fehlermeldung erscheint in der Kommandozeilenausgabe, wenn die Datei ein Zeichen enthält, das nicht zu ASCII gehört und die Datei nicht in UTF-8-Kodierung gespeichert wurd. Sie auch Abschnitt “Zeichenkodierung” in *Notationsreferenz*.

Warnung über absteigende staff affinities

Diese Warnung erscheint, wenn keine Notensysteme in der Ausgabe vorhanden sind, wenn etwa nur `ChordName`-Kontext und `Lyrics`-Kontext in einem Liedblatt vorhanden sind. Die Warnungen können vermieden werden, indem man einen der Kontexte als System erscheinen lässt, indem man ihm zu Beginn hinzufügt:

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

Zu Einzelheiten siehe „Abstand von Nicht-Notensystem-Zeilen“ in Abschnitt “Vertikale Abstände innerhalb eines Systems” in *Notationsreferenz*.

2 Dateien mit convert-ly aktualisieren

Die Eingabesyntax von LilyPond wird immer wieder verändert um Dinge zu vereinfachen oder verschiedene Verbesserungen und Entwicklungen einzubringen. Ein Nebeneffekt davon ist jedoch, dass LilyPond älter Eingabdateien oft nicht mehr richtig bearbeiten kann. Um dieses Problem zu umgehen, kann das Programm `convert-ly` benutzt werden, welches die meisten Syntaxveränderungen zwischen unterschiedlichen LilyPond-Versionen beherrscht.

2.1 Warum verändert sich die Syntax?

Die LilyPond-Eingabesyntax verändert sich von Zeit zu Zeit. Wenn das Programm LilyPond verbessert wird, wird auch die Syntax (die Eingabesprache) entsprechend angepasst. Manche Änderungen machen den Eingabetext leichter zum Schreiben und zum Lesen, andere implementieren neue Eigenschaften in LilyPond.

Beispielsweise alle `\paper`- und `\layout`-Eigenschaftsnamen sollen in der Form `erstens-zweitens-drittens` geschrieben werden. In der Version 2.11.60 bemerkten wir jedoch, dass die `printallheaders`-Eigenschaft sich nicht an diese Regel hielt. Sollten wir das jetzt lassen (womit neue Benutzer verwirrt werden, weil die Eingabe nicht logisch ist), oder sollten wir es ändern (womit wir alte Benutzer mit ihren schon geschriebenen Partituren ärgern)? In diesem Fall haben wir uns entschieden, den Namen in `print-all-headers` zu ändern. Zum Glück kann diese Änderung mit dem `convert-ly`-Programm automatisch vorgenommen werden.

Leidern kann `convert-ly` nicht mit allen Syntax-Änderungen umgehen. Beispielsweise wurden in LilyPond 2.4 und früher Akzente für verschiedene Sprachen mit LaTeX-Befehlen eingegeben – beispielsweise Änderung wurde geschrieben `\"Anderung`. Ab Version 2.6 jedoch muss man Akzente und Sonderzeichen dirket als UTF-8-Zeichen notieren. `convert-ly` kann nicht alle LaTeX-Zeichen umwandeln, sodass man das manuell übernehmen muss.

2.2 convert-ly aufrufen

`convert-ly` benutzt den Befehl `\version` mit Angabe der Versionsnummer der ursprünglichen LilyPond-Version. In den meisten Fällen genügt es, einfach auf der Kommandozeile

```
convert-ly -e meineDatei.ly
```

im Verzeichnis, in welchem die Datei liegt, aufzurufen. Hierdurch wird `meineDatei.ly` direkt aktualisiert und das Original nach `meineDatei.ly~` gesichert.

> **Achtung:** `convert-ly` konvertiert immer bis zur letzten Syntax-Änderung, die das Programm beherrscht. Das heißt, dass die `\version`-Nummer, die nach der Konversion in der Datei steht, normalerweise niedriger ist als die Version von `convert-ly` selbst.

Um alle Dateien in einem Verzeichnis zu konvertieren, schreibt man auf der Kommandozeile:

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

Man kann auch einen neuen Namen für die konvertierte Datei angeben, sodass die originale Datei unverändert bleibt. Dazu schreibt man auf der Kommandozeile

```
convert-ly meineDatei.ly > meineneueDatei.ly
```

Das Programm gibt die Versionsnummern für alle Versione aus, für die eine Konversion durchgeführt wurde. Wenn keine Versionsnummern ausgegeben werden, ist die Datei aktuell.

MacOS X-Benutzer können die Befehle unter dem Menü-Eintrag `Compile > Update syntax` ausführen.

Windows-Benutzer sollten diese Befehle auf der Kommandozeile (Eingabeaufforderung), die sich normalerweise unter Start > Zubehör > Eingabeaufforderung findet.

2.3 Optionen auf der Kommandozeile für convert-ly

Das Programm wird folgendermaßen aufgerufen:

```
convert-ly [Option]... Dateiname...
```

Folgende Optionen können benutzt werden:

`-e,--edit`
: Die Konvertierung direkt am Original vornehmen, sodass es direkt verändert wird.

`-f,--from=`*von-Versionsnummer*
: Stellt die Versionsnummer ein, ab welcher die Konvertierung begonnen werden soll. Wenn die Option nicht benutzt wird, rät convert-ly die Versionsnummer anhand des \version-Eintrags in der Datei. Beispielsweise `--from=2.10.25`

`-n,--no-version`
: Normalerweise fügt convert-ly einen \version-Eintrag zu der konvertierten Datei hinzu. Mit dieser Option wird das unterdrückt.

`-s, --show-rules`
: Zeige alle bekannten Konversionen und beende.

`--to=`*bis-Versionsnummer*
: Die Zielversion der Konversion setzen. Standard ist die letzte mögliche Version, die das Programm beherrscht. Beispielsweise `--to=2.12.2`

`-h, --help`
: Zeigt Hilfe zur Benutzung.

Um LilyPond-Schnipsel in texinfo-Dateien zu aktualisieren, kann

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

benutzt werden.

Um sich die Änderungen der LilyPond-Syntax zwischen zwei Versionen anzeigen zu lassen, schreibt man

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

2.4 Probleme mit convert-ly

Wenn man convert-ly auf einer Eingabeaufforderung unter Windows mit einer Datei benutzt, die Leerzeichen im Dateinamen oder Pfad hat, muss der gesamte Dateiname mit drei (!) doppelten Anführungszeichen umgeben werden:

```
convert-ly """D:/My Scores/Ode.ly""" > "D:/My Scores/new Ode.ly"
```

Wenn der einfache convert-ly -e *.ly-Befehl nicht funktioniert, weil die ausgeschriebene Kommandozeile zu lang wird, kann man convert-ly auch als Loop wiederholt laufen lassen. Dieses Beispiel für UNIX konvertiert alle -ly-Dateien im aktuellen Verzeichnis:

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

Für die Windows-Eingabeaufforderung lautet der entsprechende Befehl:

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

Nicht alle Syntax-Änderungen werden konvertiert. Nur eine Ausgabeoption kann angegeben werden. Automatische Aktualisierung von Scheme- und LilyPond Scheme-Code ist eher unwahrscheinlich, sehr wahrscheinlich muss hier manuell aktualisiert werden.

2.5 Manuelle Konversion

Theoretisch könnte ein Programm wie convert-ly alle möglichen Syntax-Änderungen berücksichtigen. Schließlich ist es auch ein Computerprogramm, das die alte und die neue Version der Notationsdatei interpretiert, so dass ein anderes Computerprogramm eine Datei in die andere verwandeln könnte.[1]

Das LilyPond-Team ist jedoch verhältnismäßig klein, sodass nicht alle Konversionen automatisch funktionieren. Unten eine Liste der bekannten Probleme:

```
1.6->2.0:
 Bezifferter Bass wird nicht immer richtig konvertiert, besonders {<
 >}.  Mats' Kommentar zu einer Lösung:
   To be able to run convert-ly
   on it, I first replaced all occurrences of '{<' to some dummy like '{#'
   and similarly I replaced '>}' with '&}'.  After the conversion, I could
   then change back from '{ #' to '{ <' and from '& }' to '> }'.
 Nicht alle Textbeschriftung wird richtig konvertiert.  In der alten Syntax
 konnte man mehrere Beschriftungen mit Klammern gruppieren, etwa:
   -#'((bold italic) "string")
   Das wird falsch konvertiert zu:
   -\markup{{\bold italic} "string"}
   anstelle von:
   -\markup{\bold \italic "string"}
2.0->2.2:
 Versteht nicht \partcombine
 Kann nicht \addlyrics => \lyricsto, sodass einige Dateien mit vielen Strophen nicht
 funktionieren
2.0->2.4:
 \magnify wird nicht nach \fontsize verändert.
    - \magnify #m => \fontsize #f, where f = 6ln(m)/ln(2)
 remove-tag wird nicht verändert.
    - \applyMusic #(remove-tag '. . .) => \keepWithTag #'. . .
 first-page-number wird nicht verändert.
    - first-page-number no => print-first-page-number = ##f
 Zeilenumbrüche in Titelköpfen werden nicht umgewandelt.
    - \\\\  als Zeilenumbruch in \header-Feldern => \markup \center-align <
      "Erste Zeile" "Zweite Zeile" >
 Crescendo und decrescendo-Endpunkte werden nicht umgewandelt.
    - \rced => \!
    - \rc => \!
2.2->2.4:
 \turnOff (benutzt in \set Staff.VoltaBracket = \turnOff) wird nicht richtig konvertiert
2.4.2->2.5.9
 \markup{ \center-align <{ ... }> } sollte konvertiert werden in:
 \markup{ \center-align {\line { ... }} }
 jetzt fehlt aber \line.
2.4->2.6
 Besondere LaTeX-Zeicehn wie $~$ in Text werden nicht in UTF-8 umgewandelt.
2.8
 \score{} muss jetzt immer mit einem musikalischen Ausdruck beginnen. Alles
```

[1] Das ist auf jeden Fall möglich für jede LilyPond-Datei, die kein Scheme beinhaltet. Wenn Scheme in der Datei verwendet wurde, enthält die Datei Turing-complete Sprache und es gibt Probleme mit dem „Halteproblem“ der Informatik.

```
andere (insbesondere \header{}) darf erst nach den Noten kommen.
```

3 lilypond-book aufrufen

Wenn Sie in ein Dokument Grafiken Ihres Musiksatzes einfügen möchten, so können Sie genauso vorgehen, wie Sie andere Grafiken einfügen würden: Die Bilder werden getrennt vom Dokument im PostScript- oder PNG-Format erstellt und können dann in LaTeX oder HTML eingefügt werden.

`lilypond-book` automatisiert diesen Prozess: Dieses Programm extrahiert Musik-Schnipsel aus Ihrem Dokument, ruft `lilypond` auf und fügt die resultierenden Bilder in Ihr Dokument ein. Die Länge der Zeilen und die Schriftgroße werden dabei Ihrem Dokument angepasst.

`lilypond-book` ist ein eigenständiges Programm und wird üblicherweise von der Kommandozeile aufgerufen. Nähere Informationen hierzu finden sich in Abschnitt 1.2 [Benutzung auf der Kommandozeile], Seite 1. Wenn Sie MacOS 10.3 oder 10.4 benutzen und Probleme mit `lilypond-book` haben, lesen Sie Abschnitt “MacOS X” in *Allgemeine Information*.

Dieses Vorgehen kann bei LaTeX, HTML, Texinfo oder DocBook Dokumenten angewendet werden.

3.1 Ein musikwissenschaftlicher Text als Beispiel

Zahlreiche Texte enthalten Musikbeispiele: musikwissenschaftliche Abhandlungen, Liederbücher oder Handbücher wie dieses. Solche Texte können händisch erzeugt werden, indem einfach die Musikbeispiele als Grafik (PostScript, PNG, GIF, etc.) im Textverarbeitungsprogramm eingefügt werden. Für HTML, LaTeX, Texinfo und DocBook Dokumente existiert jedoch ein Weg, dies automatisiert durchzuführen.

Das Programm `lilypond-book` extrahiert die Musikfragmente aus dem Dokument, formatiert sie automatisiert in eine Grafik und fügt die resultierenden Notenbeispiele dann wieder in das Dokument ein. Dies soll hier an einem einfachen LaTeX-Beispiel verdeutlicht werden. Das Beispiel selbst enthält schon Erklärungen, sodass wir es hier nicht weiter diskutieren müssen.

Eingabe

```
\documentclass[a4paper]{article}

\begin{document}

Dokumente für \verb+lilypond-book+ können Musik und Text nach Belieben
kombinieren. Zum Beispiel:

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

Optionen für \verb+lilypond+ werden dabei in eckige Klammern gesetzt.

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

Größere Beispiele können auch in einer eigenständigen Datei gespeichert
und dann mit \verb+\lilypondfile+ eingebunden werden.
```

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

(Falls nötig kann screech-boink.ly durch eine beliebige andere .ly
Datei im selben Verzeichnis wie diese Datei ersetzt werden.)

\end{document}
```

Verarbeitung

Speichern Sie den obigen LaTeX-Quellcode in eine Datei '`lilybook.lytex`' und führen Sie dann in der Kommandozeile folgende Befehle aus:

```
lilypond-book --output=out --pdf lilybook.lytex
lilypond-book (GNU LilyPond) 2.13.39

Reading lilybook.lytex...
..(viele Ausgabezeilen entfernt)..
Compiling lilybook.tex...
cd out
pdflatex lilybook
..(viele Ausgabezeilen entfernt)..
xpdf lilybook
(Ersetzen Sie xpdf durch Ihren PDF-Betrachter)
```
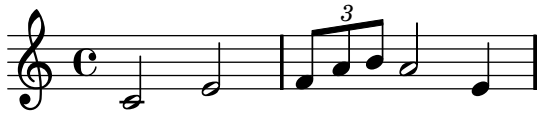
Die Ausführung von `lilypond-book` und `latex` erzeugt zahlreiche temporäre Dateien, die das Arbeitsverzeichnis unnötig vollstopfen würden. Daher empfiehlt sich die Benutzung der `--output=`*`dir`* Option, wodurch die Dateien im Unterverzeichnis '`dir`' erzeugt werden.

Das Endresultat des obigen LaTeX-Beispiels ist im nächsten Abschnitt zu sehen.[1]

[1] Da dieses Handbuch mit Texinfo erzeugt wurde, kann sich das Aussehen des Beispiels leicht von dem mit LaTeX erzeugten unterscheiden.
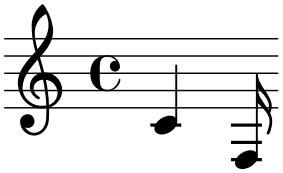
Ausgabe

Dokumente für `lilypond-book` können Musik und Text nach Belieben kombinieren. Zum Beispiel:

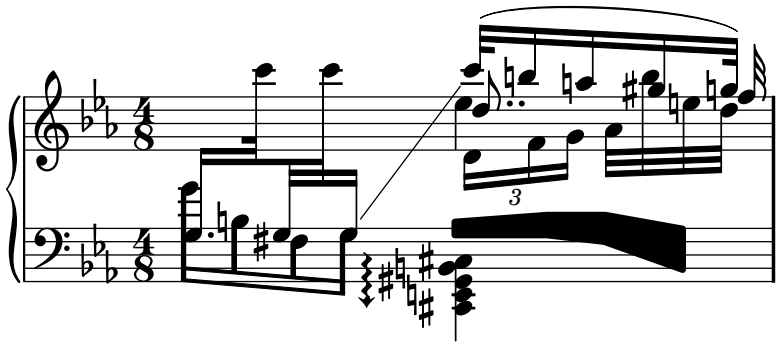


Optionen für `lilypond` werden dabei in eckige Klammern gesetzt.

```
c'4 f16
```



Größere Beispiele können auch in einer eigenständigen Datei gespeichert und dann mit `\lilypondfile` eingebunden werden.



3.2 Noten in Text integrieren

In diesem Abschnitt soll die Integration von LilyPond mit den verschiedenen Dateiformaten detailliert erläutert werden.

3.2.1 LaTeX

LaTeX ist der de-facto Standard zur Publikation von wissenschaftlichen Texten in Naturwissenschaft und Technik. Es basiert auf dem Schriftsetzer TeX, der die bestmögliche Typographie erzeugt.

Siehe die *LaTeX2e-Kurzbeschreibung* für eine Einführung in die Benutzung von LaTeX.

Musikbeispiele können eingegeben werden als

```
\begin{lilypond}[Optionen,kommen,hierhin]
  IHR LILYPOND-QUELLCODE
\end{lilypond}
```

oder

```
\lilypondfile[Optionen,kommen,hier]{Dateiname}
```

oder
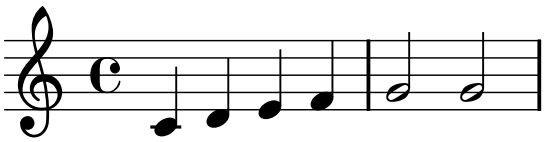
```
\lilypond{ IHR LILYPOND-QUELLCODE }
```

Zusätzlich kann mit `\lilypondversion` die benutzte Versionsnummer von LilyPond angezeigt werden. Der Aufruf von `lilypond-book` liefert eine Datei, die dann mit LaTeX weiter verarbeitet werden kann.

Dies soll hier an einigen Beispielen gezeigt werden. Die `lilypond`-Umgebung

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

erzeugt



Die Kurzversion

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

erzeugt



Innerhalb des `\lilypond{}` Befehls dürfen keine geschwungenen Klammern `{` oder `}` vorkommen, weshalb dieser Befehl nur mit der `fragment`-Option Sinn macht.

Die Standardzeilenlänge wird bestimmt, indem die Angaben in der Dokumentpräambel, also dem Teil der LaTeX Datei vor dem `\begin{document}`, analysiert werden. Der `lilypond-book`-Befehl sendet diese Angaben an LaTeX, um herauszufinden, wie breit der Text tatsächlich ist. Die Breite der Notenzeilen wird dann an diese Textbreite angepasst. Ein derartig heuristischer Algorithmus kann natürlich auch versagen, wobei man in diesem Fall die Breite auch explizit durch die `line-width` Option des `\lilypond{}` oder `\begin{lilypond}` Befehls angeben kann.

Jedes Musikbeispiele ruft die folgenden Makros auf, wenn sie vom Benutzer definiert wurden:

- `\preLilyPondExample` – wird vor der Musik aufgerufen,

- `\postLilyPondExample` – wird nach der Musik aufgerufen,
- `\betweenLilyPondSystem[1]` – wird zwischen den einzelnen Systemen aufgerufen, wenn `lilypond-book` das Beispiel in verschiedene PostScript Dateien getrennt hat. Dieser LaTeX-Befehl muss so definiert werden, dass er genau ein Argument erhält, nämlich die Zahl der bereits in LaTeX eingefügten Dateien dieses Beispiels. Als Standard wird einfach ein `\linebreak` eingefügt.

Ausgewählte Schnipsel

Manchmal ist es nötig, Musikelemente wie Halte- oder Bindebögen so darzustellen, als ob sie am Ende des Musikausschnittes noch weitergehen würden. Eine solche Ausgabe kann erreicht werden, indem ein Zeilenumbruch in die Notenzeile eingefügt wird und die Ausgabe der folgenden Notenzeile unterdrückt wird.

In LaTeX wird dazu der Befehl `\betweenLilyPondSystem` einfach derartig programmiert, dass die Ausgabe der einzelnen Notensysteme abgebrochen wird, sobald die gewünschte Anzahl an Systemen erreicht ist. Da `\betweenLilyPondSystem` zum ersten Mal nach dem ersten System aufgerufen wird, ist die Ausgabe nur eines Systems trivial.

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

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

Um eine größere Zahl an System nötig, dann muss dementsprechend eine TeX-Bedingung vor dem `\endinput` benutzt werden:

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

Dieses Beispiel bricht nach genau zwei ausgegebenen Notenzeilen ab. Für eine andere Anzahl braucht nur ‚2' durch die entsprechende Anzahl ersetzt werden.

Die Definition von `\betweenLilyPondSystem` bleibt gültig, bis TeX die aktuelle Umgebung in LaTeX verlässt oder der Befehl durch eine neue Definition überschrieben wird. Dies kann etwa folgendermaßen in der LaTeX-Datei geschehen:

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

Obige Definition von `\betweenLilyPondSystem` kann durch die Definition eines TeX-Makros auch verallgemeinert werden,

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

wobei diesem Makro `\onlyFirstNSystems` einfach die Anzahl der gewünschten Systeme übergeben wird:

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

Siehe auch

`lilypond-book` stellt auch zahlreiche Kommandozeilen-Optionen zur Verfügung. Für eine Liste dieser Optionen und andere hilfreiche Details zur Verarbeitung von LaTeX-Dokumenten, siehe Abschnitt 3.4 [lilypond-book aufrufen], Seite 26.

3.2.2 Texinfo

Texinfo ist das Standard-Dokumentationsformat des GNU Projekts. Ein Beispiel für ein Dokument im Texinfo Format ist dieses Handbuch, wobei die HTML-, PDF- und Info-Versionen alle aus demselben Texinfo Dokument erzeugt werden.

In der Eingabedatei wir Musik eingegeben als

```
@lilypond[Optionen,kommen,hier]
  IHR LILYPOND-QUELLCODE
@end lilypond
```

oder

```
@lilypond[Optionen,kommen,hier]{ IHR LILYPOND-QUELLCODE }
```

oder

```
@lilypondfile[Optionen,kommen,hier]{Dateiname}
```
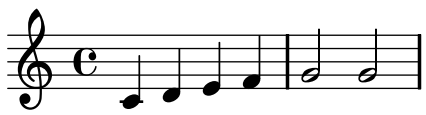
Zusätzlich kann mit `@lilypondversion` die aktuelle Versionsnummer von LilyPond angezeigt werden.

Wenn `lilypond-book` eine derartige Datei verarbeitet, wird eine Texinfo-Datei mit der Erweiterung '`.texi`' erzeugt, die `@image` Befehle für die Ausgabe nach HTML, Info und PDF enthält. `lilypond-book` erzeugt die entsprechenden Grafiken der Musikbeispiele im EPS- und PDF-Format für die Ausgabe nach PDF und im PNG-Format für die Ausgabe nach HTML und Info.

Hier sollen zwei einfache Beispiele gezeigt werden. Eine `lilypond` Umgebung

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

erzeugt



Die Kurzversion

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

erzeugt
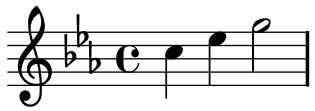


Im Gegensatz zu LaTeX erzeugt `@lilypond{...}` allerdings keine Grafik im Fließtext, sondern setzt sie immer in einen eigenen Absatz.

3.2.3 HTML

Noten werden eingegeben als

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

`lilypond-book` erzeugt dann daraus eine HTML-Datei mit den entsprechenden `<image>` Tags für die Musikbeispiele in jeweils einem eigenen Absatz.



Für Grafiken im Fließtext kann `<lilypond ... />` benutzt werden, wobei die Optionen durch einen Doppelpunkt von der Musik getrennt angegeben werden.

```
Musik <lilypond relative=2: a b c/> in derselben Zeile.
```

Um Dateien mit Musik einzubinden, kann folgendermaßen vorgegangen werden:

```
<lilypondfile Option1 Option2 ...>Dateiname</lilypondfile>
```

Zusätzlich gibt `<lilypondversion/>` die aktuelle Versionsnummer von LilyPond aus.

3.2.4 DocBook

Bei der Einbindung von Musik im LilyPond-Format in DocBook soll die Konformität unseres DocBook Dokuments erhalten bleiben und damit die Bearbeiten mit DocBook-Editoren sowie die Validierung weiter möglich bleiben. Aus diesem Grund werden in DocBook keine eigenen Tags wie in HTML benutzt, sondern die von den vorhandenen DocBook-Elementen vorgegebenen Konventionen entsprechend benützt.

Definitionen

Für die Einbindung von LilyPond Code werden in allen Fällen die `mediaobject` und `inlinemediaobject` Elemente benutzt, die unsere Beispiele in einem eigenen Absatz oder im Fließtext einfügen. Die Optionen zur Formatierung mit LilyPond werden dabei in der `role` Eigenschaft des innersten Elements angegeben, wie im nächsten Abschnitt gezeigt wird. Die DocBook Datei, die dann von `lilypond-book` verarbeitet wird, sollte der Klarheit halber die Dateierweiterung '`.lyxml`' (jedenfalls nicht '`.xml`') besitzen.

Eine LilyPond-Datei einfügen

Dies ist der einfachste Fall: Die LilyPond-Datei besitzt die Erweiterung '`.ly`' und wird einfach als `imageobject` eingebettet:

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

Für das äußerste Element kann je nach Bedarf `mediaobject` oder `inlinemediaobject` benutzt werden.

LilyPond-Code einfügen

Die Einbindung von LilyPond-Code direkt in der DocBook-Datei ist durch die Benutzung von `programlisting` möglich, wobei die Sprache auf `lilypond` gesetzt wird:

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

Das äußerste Element ist also `mediaobject` oder `inlinemediaobject`, welches ein `textobject` mit dem `programlisting` enthält.

Ein DocBook-Dokument übersetzen

Der Aufruf von `lilypond-book` mit der '`.lyxml`'-Datei erzeugt ein gültiges DocBook-Dokument mit der Erweiterung '`.xml`', welches normal weiterverarbeitet werden kann. Bei Benutzung von dblatex wird daraus automatisch eine PDF-Datei erzeugt. Für die Erzeugung von HTML (HTML Hilfe, JavaHelp, etc.) können die offiziellen DocBook XSL-Stylesheets benutzt werden. Eventuell müssen dafür allerdings kleinere Anpassungen vorgenommen werden.

3.3 Die Musikfragment-Optionen

Im Folgenden meint ‚LilyPond-Befehl' einen beliebigen in den vorgehenden Abschnitten beschriebenen Befehl, der ein LilyPond-Fragment in eine Datei einfügt und von `lilypond-book` verarbeitet wird. Der Einfachheit halber werden hier alle LilyPond-Befehle in der Syntax von LaTeX dargestellt.

Zu beachten ist, dass die Optionen eines LilyPond-Befehls von links nach rechts verarbeitet werden. Wenn eine Option also mehrfach angegeben wird, wird nur die letzte benutzt.

Die folgenden Optionen können für LilyPond-Befehle benutzt werden:

`staffsize=`*ht*
: Setzt die Höhe einer Notenzeile auf *ht*, angegeben in Punkten.

`ragged-right`
: Erzeugt Notenzeilen im Flattersatz mit natürlichem Abstand der Noten. In anderen Worten: `ragged-right = ##t` wird in das Musikfragment eingefügt. Dies ist die Standardeinstellung für den `\lilypond{}`-Befehl, wenn die Option `line-width` nicht angegeben wird. Ebenso ist dies die Standardeinstellung für die `lilypond`-Umgebung, wenn die Option `fragment`, aber keine Zeilenlänge explizit angegeben ist.

`noragged-right`
: Streckt Musikfragmente mit nur einer Notenzeile auf die volle Breite, es wird also `ragged-right = ##f` in das Musikfragment eingefügt.

`line-width`
`line-width=`*Breite*`\`*Einheit*
: Setzt die Breite der Notenzeilen auf *Breite*, gemessen in Vielfachen der *Einheit*. Als Einheit können die folgenden Zeichenfolgen angegeben werden: `cm`, `mm`, `in` oder `pt`. Diese Option hat nur Einfluss auf die Breite von Notenzeilen und Text im Musikfragment, nicht jedoch auf den restlichen Text des Dokuments.

  Wird diese Option ohne einen Wert angegeben, wird die Zeilenbreite auf einen Standardwert gesetzt, der durch einen heuristischen Algorithmus bestimmt wird.

  Wenn die `line-width`-Option nicht angegeben wird, versucht `lilypond-book` einen geeigneten Standardwert für alle `lilypond`-Umgebungen zu finden, die die `ragged-right`-Option nicht benutzen.

`notime`
: Verhindert die Ausgabe der Taktangabe am Anfang des Fragments und schaltet Taktstriche und alle Taktangaben im Fragment ab.

`fragment`
: Bewirkt, dass `lilypond-book` Standardcode um das Fragment herum einfügt, sodass z. B.

  ```
  c'4
  ```

  ohne `\layout`, `\score`, etc. eingegeben werden kann.

`nofragment`
: Verhindert das Einfügen von Standardcode um das Fragment herum. Dies ist die Standardeinstellung, insofern ist diese Option im Normalfall unnötig.

`indent=`*`Einzug`*`\`*`Einheit`

Setzt den Einzug des ersten Notensystems auf *Einzug*, gemessen in Vielfachen der *Einheit*. Als Einheit können die folgenden Zeichenfolgen angegeben werden: `cm`, `mm`, `in` oder `pt`. Diese Option hat nur Einfluss auf den Einzug von Notenzeilen und Text im Musikfragment, nicht jedoch auf den restlichen Text des Dokuments.

`noindent` Setzt den Einzug des ersten Notensystems auf 0. Diese Option hat nur Einfluss auf den Einzug von Notenzeilen und Text im Musikfragment, nicht jedoch auf den restlichen Text des Dokuments. Dies ist die Standardeinstellung, insofern ist diese Option im Normalfall unnötig.

`quote` Verringert die Zeilenlänge des Musikfragments um $2 * 0.4$ in und setzt das Fragment in einen Zitat-Block. Der Wert von ‚0.4 in‘ kann durch die `exampleindent` Option angepasst werden.

`exampleindent`

Setzt den Betrag, um den das Fragment bei Benutzung der `quote` Option eingerückt wird.

`relative`
`relative=`*`n`*

Benutzt relative Oktavenbezeichnungen. Standardmäßig werden Noten relativ zum mittleren C angegeben. Das optionale ganzzahlige Argument gibt die Oktave der ersten Note an, wobei die Standardeinstellung von `1` das mittlere C bedeutet. Die `relative` Option macht nur Sinn in Verbindung mit der `fragment` Option, weshalb `fragment` automatisch durch die Angabe der `relative` Option impliziert wird. Eine explizite Angabe der `(no)fragment` Option hat keinen Effekt.

LilyPond benutzt zur Erzeugung seiner eigenen Dokumentation ebenfalls `lilypond-book`. Zu diesem Zweck stehen noch zahlreiche spezialisierte Optionen zur Verfügung:

`verbatim` Der LilyPond-Code im LilyPond-Kommando wird zum einen benutzt, um das Musikfragment in eine Grafik mit schönem Notensatz zu konvertieren, andererseits aber auch wörtlich in das Dokument eingefügt. Dies geschieht in einem ‚verbatim‘-Block, gefolgt vom Text einer möglicherweise angegebenen `intertext` Option[1] und der Grafik des tatsächlichen Notensatzes. Diese Option funktioniert nur fehlerhaft, wenn `\lilypond{}` im Fließtext benutzt wird.

Wird `verbatim` in Verbindung mit einem `lilypondfile`-Kommando benutzt, so ist es auch möglich, nur ein Teil der Datei wörtlich einfügen zu lassen: Wenn die eingebundene LilyPond-Datei ein Kommentar mit dem Inhalt ‘`begin verbatim`’ (ohne Anführungszeichen) enthält, wird nur der Dateiinhalt ab dieser Position eingefügt. Enthält die Datei mehrere solche Kommentare, wirkt nur das letzte. Analog wird nur der Dateiinhalt bis zu einem etwaigen Kommentar mit dem Inhalt ‘`end verbatim`’ eingefügt. Im folgenden Beispiel wird das gesamte Musik für die Erzeugung der Grafik im relativen Oktavenmodus interpretiert, der wörtlich in das Dokument kopierte LilyPond-Code zeigt den `relative`-Befehl jedoch nicht.

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

erzeugt ein Zitat der Form

```
  c4 e2 g4
  f2 e
```

[1] Die `intertext` Option ist noch nicht implementiert.

Wenn Kommentare und Variablen im Zitat, aber nicht im Quelltext übersetzt werden sollen, kann die Umgebungsvariable LYDOC_LOCALEDIR auf einen Verzeichnispfad gesetzt werden. Das Verzeichnis sollte einen Baum an '.mo'-Nachrichtenkatalogen beinhalten mit lilypond-doc als Domain.

addversion

(Nur innerhalb von Texinfo-Dateien.) Stellt \version @w{"@version{}"} an den Beginn des Fragments der Ausgabe mit verbatim.

texidoc (Nur innerhalb von Texinfo-Dateien.) Wird lilypond mit der Kommandozeilenoption '--header=texidoc' für eine Datei 'foo.ly' und enthält die Datei ein texidoc-Feld im \header-Block, so wird dessen Inhalt in die Datei 'foo.texidoc' ausgegeben. Die texidoc Option veranlasst lilypond-book, den Inhalt dieser '.texidoc' Dateien innerhalb eines Dokumentationsblocks direkt vor dem Musikfragment in das aktuelle Dokument einzufügen.

Enthält also die Datei 'foo.ly' etwa den LilyPond-Code

```
\header {
  texidoc = "Dieses Beispiel zeigt eine einzelne Note."
}
{ c'4 }
```

und das Texinfo-Dokument 'text.texinfo'

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

so liefert der folgende Aufruf von lilypond-book das gewünschte Ergebnis:

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

Die meisten Test-Dateien (im 'input/' Verzeichnis von LilyPond) sind kleine '.ly' Dateien von genau dieser Form.

Auch die Übersetzung dieser zusätzlichen Kommentare ist möglich: Dazu muss das Texinfo-Dokument den Befehl @documentlanguage *LANG* und der \header Block in der Datei 'foo.ly' die Übersetzung im Feld texidoc*LANG* enthalten. Wird nun lilypond mit der Option '--header=texidoc*LANG*' aufgerufen, so wird der Inhalt der Datei 'foo.texidoc*LANG*' anstelle von 'foo.texidoc' eingefügt.

lilyquote

(Nur innerhalb von Texinfo-Dateien.) Diese Option wirkt wie die quote Option, außer dass nur das Musikfragment (und natürlich optional der gesamte LilyPond-Code bei Benutzung von verbatim) in den Zitatblock eingefügt wird. Dies ist nützlich, wenn Sie ein Musikfragment zitieren möchten, nicht aber dessen texidoc Dokumentationsblock.

doctitle (Nur innerhalb von Texinfo-Dateien.) Diese Option wirkt ähnlich wie die texidoc Option: Wenn lilypond mit der Option '--header=doctitle' aufgerufen wird und die Eingabedatei 'foo.ly' ein Feld doctitle im \header-Block enthält, wird dessen Wert in die Datei 'foo.doctitle' geschrieben. Wird die doctitle Option für ein Musikfragment benutzt, so wird der Inhalt dieser Datei, der eine einzelne Textzeile sein sollte, im Texinfo-Dokument als @lydoctitle *Text* eingefügt. @lydoctitle muss allerdings in Ihrem Texinfo-Dokument als Makro selbst definiert werden. Die Übersetzung funktioniert völlig analog zu texidoc.

nogettext

Nur für Texinfo-Ausgabe: Kommentare und Variablenbezeichnungen im zitierten Quelltext des Schnipsel werden nicht übersetzt.

`printfilename`
: Wenn eine LilyPond-Datei mittels `\lilypondfile` und dieser Option eingebunden wird, wird der Dateiname (ohne die Pfadangabe) unmittelbar vor dem Musikfragment ausgegeben. In HTML-Dateien ist er außerdem ein Link auf die LilyPond-Datei. Nur der eigentliche Name der Datei wird ausgegeben, der Pfad wird also nicht mit angezeigt.

3.4 `lilypond-book` aufrufen

`lilypond-book` erzeugt abhängig vom Ausgabeformat eine Datei mit einer der folgenden Dateierweiterungen: '`.tex`', '`.texi`', '`.html`' oder '`.xml`'. Alle '`.tex`', '`.texi`' und '`.xml`' Dateien müssen noch mit den entsprechenden Programmen (LaTeX, DocBook, etc.) weiter verarbeitet werden, um druckfähige Dateien zu erhalten.

Formatabhängige Anweisungen

LaTeX

Es existieren zwei Methoden, Ihr LaTeX-Dokument weiter zu verarbeiten, um zu einer druck- oder publikationsfähigen Datei zu gelangen: Zum einen die direkte Erzeugung einer PDF-Datei mit PDFLaTeX, zum anderen die Erzeugung einer DVI daraus einer PostScript-Datei mittels LaTeX und einem DVI-nach-PostScript Konverters wie `dvips`. Die erste Methode ist einfacher und daher empfehlenswert.[2] Welche Methode auch immer benutzt wird, die Konvertierung zwischen PostScript und PDF kann leicht mit Hilfsprogrammen wie `ps2pdf` und `pdf2ps` (aus dem Ghostscript-Paket) erfolgen.

Um eine PDF-Datei mittels PDFLaTeX zu erzeugen, kann folgendermaßen vorgegangen werden:

```
lilypond-book --pdf Ihre_Datei.pdftex
pdflatex Ihre_Datei.tex
```

Um eine PDF-Datei mittels LaTeX/`dvips`/`ps2pdf` zu erhalten, sind folgende Befehle nötig:

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

Die '`.dvi`'-Datei, die beim Aufruf von `latex` erzeugt wird, scheint keine Notenköpfe zu enthalten, was in Ordnung ist. Wenn Sie die Datei wie beschrieben weiter verarbeiten, erscheinen die Notenköpfe korrekt in den '`.ps`' und '`.pdf`' Dateien.

Der Aufruf von `dvips` kann einige Warnungen über fehlende Schriftarten ausgeben. Auch dies ist in Ordnung und kann ignoriert werden.

Wenn Sie in der LaTeX-Datei das Papierformat auf Querformat eingestellt haben, vergessen Sie nicht die `-t landscape`-Option beim Aufruf von `dvips`.

Texinfo

Um ein von `lilypond-book` erzeugtes Texinfo-Dokument zu verarbeiten, gehen Sie wie für alle anderen Texinfo-Dokumente vor: Rufen Sie – abhängig vom gewünschten Ausgabeformat – eines der Programme `texi2pdf`, `texi2dvi`, `makeinfo` oder `texi2html` auf.

Die Dokumentation von Texinfo liefert dazu nähere Informationen.

[2] Manchmal kann eine Datei entweder von PDFLaTeX oder von LaTeX nicht korrekt verarbeitet werden, weshalb hier beide Methoden beschrieben werden.

Optionen auf der Kommandozeile

lilypond-book unterstützt die folgenden Kommandozeilenoptionen:

-f *Format*
--format=*Format*
: Gibt das Format des Eingabedokuments an: html, latex, texi (Standardeinstellung), texi-html oder docbook. Ist diese Option nicht angegeben, versucht lilypond-book das Format anhand des Dateinamens zu bestimmen. Im Moment bedeutet texi praktisch dasselbe wie texi-html.

-F *Filter*
--filter=*Filter*
: Leitet die Musikfragmente durch das Programm *filter* anstatt sie mit Lilypond zu einer Grafik zu verarbeiten. '--filter' und '--process' kann nicht gleichzeitig benutzt werden. Beispielaufruf:

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

-h
--help
: Gibt eine kurze Hilfemeldung aus.

-I *Pfad*
--include=*Pfad*
: Fügt *Pfad* zu den Include-Pfaden hinzu. lilypond-book sucht auch in allen Include-Pfaden nach bereits erstellten Grafiken für die Musikfragmente. Wird das aktuelle Fragment gefunden und hat sich seit der letzten Erstellung nicht geändert, wird es nicht erneut erzeugt. Bei manchen der Programme zur Weiterverarbeitung wie etwa makeinfo oder latex muss dieselbe -I *Pfad* Option angegeben werden, damit das entsprechende Programm die Grafiken ebenso findet.

-o *Verzeichnis*
--output=*Verzeichnis*
: Erzeugt die Ausgabedateien in *Verzeichnis*. Der Aufruf von lilypond-book erzeugt zahlreiche kleine Dateien, die von LilyPond, latex, makeinfo etc. dann weiter benützt werden. Um zu vermeiden, dass das Quellenverzeichnis durch diese Dateien unübersichtlich wird, kann die '--output' Option benutzt werden. Vor dem Aufruf von latex oder makeinfo sollten Sie in dieses Verzeichnis wechseln.

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

--skip-lily-check
: Nicht mit einer Fehlermeldung abbrechen, wenn keine Ausgabe von LilyPond gefunden wird. Dies wird benutzt für Dokumentation ohne Grafiken.

--skip-png-check
: Nicht mit einer Fehlermeldung abbrechen, wenn für die EPS-Dateien keine PNG-Grafiken gefunden werden. Dies wird benutzt für Dokumentation ohne Grafiken.

--lily-output-dir=*Verzeichnis*
: Schreibt 'lily-XXX' Dateien nach *Verzeichnis* und erzeugt im mit --output angegebenen Verzeichnis Verknüpfungen darauf. Diese Option ist nützlich, um

Zeit zu sparen, wenn Dokumente in verschiedenen Verzeichnissen viele identische Musikfragmente enthalten.

`--info-images-dir=`*Verzeichnis*
: Formatiert die Texinfo-Ausgabe dergestalt, dass Info in *Verzeichnis* nach den Grafiken zu den Musikfragmenten sucht.

`--latex-program=`*Programm*
: Führt `Programm` anstelle von `latex` aus. Dies ist nützlichen, wenn das Dokument mit einer anderen LaTeX-Variante wie etwa `xelatex` verarbeitet werden soll.

`--left-padding=`*Einrückung*
: Fügt *Einrückung* als zusätzlichen Einzug in die EPS-Box ein. *Einrückung* wird in Millimetern angegeben, die Standardeinstellung ist 3.0 Millimeter. Diese Option kann benutzt werden, wenn die Notenzeilen über den Rand des Dokuments hinausstehen.

  Die Breite eines eng ausgeschnittenen Notensystems kann variieren aufgrund von Notationselementen, die über den linken Rand hinausstehen, wie etwa Taktzahlen und Bezeichnungen der Instrumente. Diese Option verkürzt die Notenzeile und verschiebt sie um denselben Betrag nach rechts.

`-P` *Befehl*
`--process=`*Befehl*
: Verarbeitet LilyPond-Fragmente mit *Befehl* anstelle des Standardbefehls `lilypond`. `--filter` und `--process` können nicht mit dieser Option benutzt werden.

`--pdf` Erzeugt PDF-Dateien mit PDFLaTeX.

`-V`
`--verbose`
: Gibt ausführliche informative Meldungen aus.

`-v`
`--version`
: Gibt die Versionsnummer aus.

Bekannte Probleme und Warnungen

Der Texinfo-Befehl `@pagesizes` wird ignoriert. Ebenso werden LaTeX-Befehle ignoriert, die den Seitenrand oder die Zeilenlänge nach der Dokumentpräambel verändern.

Nur der erste `\score`-Block eines LilyPond-Fragments wird verarbeitet.

3.5 Dateiendungen

Für die Eingabedatei kann zwar jede beliebige Dateinamenserweiterung benutzt werden, allerdings muss bei Verwendung einer nicht bekannten Erweiterung das Ausgabeformat explizit an `lilypond-book` angegeben werden. Details dazu finden sich im Abschnitt Abschnitt 3.4 [lilypond-book aufrufen], Seite 26. Wird eine bekannte Erweiterung benutzt, wählt `lilypond-book` automatisch das richtige Ausgabeformat basierend auf der Erweiterung der Eingabedatei:

Erweiterung	**Ausgabeformat**
'`.html`'	HTML
'`.itely`'	Texinfo
'`.latex`'	LaTeX
'`.lytex`'	LaTeX
'`.lyxml`'	DocBook
'`.tely`'	Texinfo

'`.tex`'	LaTeX
'`.texi`'	Texinfo
'`.texinfo`'	Texinfo
'`.xml`'	HTML

Wird dieselbe Erweiterung für die Eingabedatei wie für die Ausgabedatei benutzt und befindet sich die Eingabedatei im aktuellen Arbeitsverzeichnis von `lilypond-book`, muss die `--output` Option für `lilypond-book` benutzt werden. Anderenfalls würde `lilypond-book` ja die Eingabedatei überschreiben, weshalb das Programm in diesem Fall mit einer Fehlermeldung wie „Fehler: Ausgabe würde Eingabedatei überschreiben; verwenden Sie –output.“ abbricht.

3.6 `lilypond-book`-Vorlagen

Diese Vorlagen sollen mit `lilypond-book` benutzt werden. Wenn Sie sich mit dem Programm nicht auskennen, lesen Sie bitte Kapitel 3 [lilypond-book], Seite 16.

3.6.1 LaTeX

Sie können LilyPond-Fragmente in ein LaTeX-Dokument einfügen:

```
\documentclass[]{article}

\begin{document}

Normaler LaTeX-Text.

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

Weiterer LaTeX-Text und Optionen in eckigen Klammern.

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

3.6.2 Texinfo

LilyPond-Fragmente können in Texinfo-Dokumente eingefügt werden: dieses gesamte Handbuch wurde in Texinfo geschrieben.

```
\input texinfo @node Top
@top

Texinfo-Text

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

Weiterer Texinfo-Text und Optionen in Klammern.
```

```
@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>
Dokumente für lilypond-book können Noten und Text frei kombinieren. Zum
Beispiel
<lilypond>
\relative c'' {
  a4 b c d
}
</lilypond>
</p>

<p>
Noch etwas LilyPond, mit Optionen:

<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.
\end{document}
```

3.7 Das Inhaltsverzeichnis flexibel einsetzen

Diese Funktionen existieren schon im OrchestraLily-Paket:

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

Um den Text flexibler behandeln zu können, bevorzugen manche Benutzer es, das Inhaltsverzeichnis aus LilyPond zu exportieren und dann mit LaTeX einzulesen.

Das Ihaltsverzeichnis (ToC) aus LilyPond exportieren

Hier wird angenommen, dass in Ihrer Partitur mehrere Sätze vorkommen, die sich alle in der selben LilyPond-Datei befinden.

```
#(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))
}
```

Das Inhaltsverzeichnis in LaTeX importieren

In der LaTeX-Datei sollte folgendes enthalten:

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

wobei `\includescore` wiefolgt definiert ist:

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

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

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


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

3.8 Alternative Methoden Text und Musik zu kombinieren

Andere Methoden, Text und Noten miteinander zu kombinieren (ohne `lilypond-book` zu benutzen) werden beschrieben in Abschnitt 4.4 [LilyPond-Ausgabe in anderen Programmen], Seite 39.

4 Externe Programme

LilyPond kann mit anderen Programmen auf verschiedene Weise interagieren.

4.1 Point and click

Point and click erlaubt es, die Noten in der Eingabedatei schnell zu finden, indem man auf sie im PDF-Programm klickt. Das erleichtert es, die Stellen zu finden, die Fehler in der Notation verursachen.

Wenn diese Funktionalität aktiv ist, fügt LilyPond Hyperlinks zur PDF-Datei hinzu. Diese Hyperlinks werden an den Webbrowser gesendet, der einen Texteditor mit dem Cursor an der richtigen Stelle öffnet.

Damit diese Kettenreaktion funktionieren kann, müssen Sie das PDF-Programm dazu anweisen, Hyperlinks zu folgen, indem das Skript '`lilypond-invoke-editor`', welches mit LilyPond kommt, verwendet wird.

Für Xpdf unter UNIX sollte folgende Zeile in der Datei '`xpdfrc`'[1] stehen:

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

Das Programm '`lilypond-invoke-editor`' ist ein kleines Hilfsprogramm. Es ruft einen Editor für besondere `textedit`-URIs# auf und einen Webbrowser für andere. Es testes die Umgebungsvariable `EDITOR` nach folgenden Mustern:

`emacs` das ruft auf

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

`gvim` das ruft auf

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

`nedit` das ruft auf

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

Die Umgebungsvariable `LYEDITOR` wird benutzt, um dieses Verhalten zu umgehen. Sie enthält die Kommandozeile, mit der der Editor aufgerufen wird, wobei `%(file)s`, `%(column)s` und `%(line)s` mit der Datei, Spalte und Zeile ersetzt wird. Die Einstellung

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

für `LYEDITOR` entspricht dem normalen Aufruf von emacsclient.

Die point-and-click-Links vergrößern die Größe des PDFs sehr stark. Um die Größe von PDFs und auch PS-Dateien zu verkleinern, kann point and click ausgeschaltet werden, indem man in der Eingabedatei

```
\pointAndClickOff
```

schreibt. Point and click kann explizit aktiviert werden mit dem Befehl

```
\pointAndClickOn
```

Alternativ können Sie point and click auch mit einer Kommandozeilenoption anschalten:

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

> **Achtung:** Sie sollten immer point and click ausschalten, wenn Sie LilyPond-Dateien verteilen wollen, damit keine Informationen über Ihre Dateistrukturen in den Dateien gespeichert werden, was ein Sicherheitsrisiko darstellen könnte.

[1] Unter UNIX findet man diese Datei entweder in '`/etc/xpdfrc`' oder als '`.xpdfrc`' im home-Verzeichnis.

4.2 Unterstützung von Texteditoren

Verschiedene Editoren unterstützen LilyPond

Emacs-Modus

Emacs hat einen '`lilypond-mode`'-Modus, der Ergänzung von Befehlen, Einrückung, Syntaxhervorhebung, die Paarung von LilyPond-Klammern, einfache Tastaturkürzel zur Übersetzung und das Lesen von LilyPond-Dokumentation mit Info unterstützt. Wenn '`lilypond-mode`' nicht auf Ihrem Computer installiert ist, siehe unten.

Ein Emacs-Modus zur Noteneingabe und zum Kompilieren mit LilyPond bfindet sich in dem Quellarchiv im '`elisp`'-Verzeichnis. Führen Sie `make install` aus, damit es nach *elispdir* installiert wird. Die Datei '`lilypond-init.el`' sollte in *Ladepfad*'`/site-start.d/`' gespeichert werden oder Ihrer '`~/.emacs`' bzw. '`~/.emacs.el`' hinzugefügt werden.

Als ein Benutzer können Sie Ihren Quellpfad (z. B. '`~/site-lisp/`') zum *Ladepfad* hinzufügen, indem Sie die (veränderte) folgende Zeile in Ihr '`~/.emacs`' kopieren:

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

Vim-Modus

Für Vim gibt es ein Dateiartplugin, einen Einzugsmodus und einen Syntaxhervorhebungsmodus für LilyPond. Um alle diese Eigenschaften zu aktivieren, erstellen (oder verändern) Sie die Datei '`$HOME/.vimrc`', sodass sie folgende Zeilen enthält:

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

Wenn LilyPond nicht in '`/usr/local/`' installiert ist, müssen Sie den Pfad anpassen. Das wird besprochen in Abschnitt "Mehr Information" in *Handbuch zum Lernen*.

Andere Editoren

Andere Editoren (sowohl Texteditoren als auch graphische Editoren) haben Unterstützung für LilyPond, aber ihre besonderen Konfigurationsdateien werden nicht mit LilyPond zusammen verteilt. Lesen Sie die entsprechende Dokumenation zu näheren Fragen. Derartige Editoren finden sich unter Abschnitt "Leichteres Editieren" in *Allgemeine Information*.

4.3 Von anderen Formaten konvertieren

LilyPond kann auch Musik aus diversen anderen Formaten importieren. Dieses Kapitel beschreibt die dazu mit LilyPond mitgelieferten Hilfsprogramme. Daneben existieren natürlich auch noch weitere Programme, die Dateien für LilyPond erstellen können, wie etwa graphische Sequenzierprogramme und XML-Konverter. Näheres dazu findet sich auf der Homepage von LilyPond.

Die im Folgenden beschriebenen Programme sind eigenständige Hilfsprogramme und werden üblicherweise von der Kommandozeile aufgerufen. Siehe Abschnitt 1.2 [Benutzung auf der Kommandozeile], Seite 1 für weitere Informationen. Wenn Sie MacOS 10.3 oder 10.4 benutzen und Probleme mit diesen Skripten (z. B. `convert-ly`) haben, lesen Sie Abschnitt "MacOS X" in *Allgemeine Information*.

Bekannte Probleme und Warnungen

Leider haben wir nicht ausreichend viele Entwickler, um all die folgenden Hilfsprogramme ständig zu warten. Wir stellen den aktuellen Stand zur Verfügung, können aber leider Fehlerberichte nur selten bearbeiten. Selbstverständlich sind Patches von Ihnen sehr willkommen!

4.3.1 `midi2ly` aufrufen

`midi2ly` übersetzt eine Typ 1 MIDI-Datei in eine Eingabedatei für LilyPond.

MIDI (Music Instrument Digital Interface) ist ein internationaler Standard für digitale Instrumente: Es spezifiziert die Verkabelung, ein serielles Protokoll und ein Dateiformat. Das MIDI-Dateiformat ist der de-facto Standard um Musik von vielen Programmen zu exportieren. Allerdings fehlen in den MIDI-Dateien viele Ausdrucks- und Artikulationszeichen. Dennoch kann MIDI vielfach nützlich sein, um Musik von einem Programm zu importieren, für das kein spezielles Hilfsprogramm den direkten Import nach LilyPond unterstützt.

`midi2ly` konvertiert die MIDI-Spuren nach Abschnitt “Staff” in *Referenz der Interna* und MIDI-Kanäle in Abschnitt “Voice” in *Referenz der Interna* Kontexte. Tonhöhen werden relativ angegeben, Tondauern nur wenn nötig.

MIDI-Dateien können auch direkt von einem digitalen Keyboard aufgenommen und dann in eine ‘`.ly`’-Datei konvertiert werden. Allerdings sind Musikinterpretationen von Menschen (aus gutem Grund!) rhythmisch nicht exakt genug um die Konvertierung von MIDI nach LY trivial zu gestalten. Wenn `midi2ly` mit Quantisierung (`-s` und `-d` Kommandozeilenoptionen) aufgerufen wird, versucht es diese Unschärfen im Zeitablauf zu korrigieren, ist allerdings nicht sonderlich gut darin. Daher können wir diese Technik leider nicht für die Konvertierung von MIDI-Aufnahmen empfehlen.

`midi2ly` wird von der Kommandozeile folgendermaßen aufgerufen:

```
midi2ly [Optionen]... MIDI-Datei
```

Unter ‚Kommandozeile‘ verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Abschnitt 4.3 [Von anderen Formaten konvertieren], Seite 35.

Die folgenden Kommandozeilenoptionen werden von `midi2ly` unterstützt:

`-a, --absolute-pitches`
: Gibt absolute Tonhöhen aus.

`-d, --duration-quant=`*LÄNGE*
: Quantisiert Tondauern zu Vielfachen von *LÄNGE*.

`-e, --explicit-durations`
: Gibt alle Tondauern explizit an.

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-k, --key=`*acc*`[:`*Moll*`]`
: Setzt die Standard-Tonart. $acc > 0$ gibt die Anzahl der Kreuze an, $acc < 0$ gibt die Anzahl der Bs der Tonart an. Eine Moll-Tonart wird durch `:1` angegeben.

`-o, --output=`*Datei*
: Die Ausgabe wird in die Datei *Datei*‘`.ly`’ geschrieben.

`-s, --start-quant=`*LÄNGE*
: Quantisiert den Beginn der Noten zu Vielfachen von *LÄNGE*.

`-t, --allow-tuplet=`*DUR*NUM/DEN*
: Erlaubt Tuplet-Dauern der Form *DUR***NUM*/*DEN*.

`-V, --verbose`
: Gibt ausführliche informative Meldungen während der Konvertierung aus.

`-v, --version`
: Gibt die Versionsnummer aus.

`-w, --warranty`
: Zeigt die Lizenzbedingungen und Urheberrechtshinweise.

`-x, --text-lyrics`
: Interpretiert alle Texte als Liedtexte.

Bekannte Probleme und Warnungen

Überlappende Noten in einem Arpeggio werden nicht korrekt dargestellt. Nur die erste Note wird eingelesen und konvertiert, die restlichen werden ignoriert. Als Abhilfe können Sie alle Noten auf dieselbe Tonlänge setzen und Phrasierungszeichen oder Pedalindikatoren hinzufügen.

4.3.2 `muscxml2ly` aufrufen

MusicXML ist ein XML-Dialekt zur Darstellung von Musiknotation.

`musicxml2ly` wandelt eine MusicXML-Datei nach LilyPond um, wobei sowohl die Noten, Artikulationszeichen, Struktur der Partitur, Liedtexte etc. einer MusicXML-Datei (im ‚part-wise'-Format) in eine .ly-Datei um.

`musicxml2ly` wird von der Kommandozeile folgendermaßen aufgerufen:

```
musicxml2ly [Optionen]... XML-Datei
```

Unter ‚Kommandozeile' verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Abschnitt 4.3 [Von anderen Formaten konvertieren], Seite 35.

Wenn als Dateiname '`-`' angegeben wird, liest `musicxml2ly` Daten direkt von der Kommandozeile ein.

Die folgenden Kommandozeilenoptionen werden von `musicxml2ly` unterstützt:

`-a, --absolute`
: Konvertiert in absolute Tonhöhen.

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-l, --language=LANG`
: Inkludiert die Sprachdatei *LANG*'`.ly`' in der Ausgabe. Wird z.B. 'deutsch' angegeben, wird '`deutsch.ly`' in die .ly-Datei eingebunden und deutsche Notenbezeichnungen benutzt.

`--lxml` Benutzt das lxml.etree Python-Paket für die Verarbeitung von XML (benötigt weniger Speicher und Rechenleistung)

`--nd --no-articulation-directions`
: Konvertiert keine Richtungsangaben (`^`, `_` oder `-`) von Artikulations- und Lautstärkebezeichnungen.

`--no-beaming`
: Konvertiert keine Informationen über die Balkensetzung aus der MusicXML-Datei. Stattdessen wird dies LilyPond überlassen.

`-o,--output=`*Dateiname*
: Die Ausgabe wird in die Datei *Dateiname*'`.ly`' geschrieben. Wird als *Dateiname* nur '`-`' angegeben, wird das Ergebnis der Konvertierung an der Kommandozeile ausgegeben. Wird diese Option nicht angegeben, so erfolgt die Ausgabe in die Datei *XML-Datei*'`.ly`'.

`-r,--relative`
: Konvertiert in relative Tonhöhen. (Standardeinstellung)

`-v, --verbose`
: Gibt ausführliche informative Meldungen während der Konvertierung aus.

`--version`
: Gibt die Versionsnummer aus.

`-z,--compressed`
: Die Eingabedatei wird als komprimierte MusicXML-Datei eingelesen. Dies ist die Standardeinstellung für Dateien mit der Erweiterung '`.mxl`'.

4.3.3 `abc2ly` aufrufen

> **Achtung:** Dieses Programm ist nicht unterstützt und kann aus kommenden LilyPond-Versionen entfernt werden.

ABC ist ein relativ einfaches ASCII-basierendes Musikformat und ist dokumentiert auf der ABC-Homepage:

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

`abc2ly` konvertiert ABC-Dateien nach LilyPond und wird von der Kommandozeile folgendermaßen aufgerufen:

`abc2ly [`*`Optionen`*`]... `*`ABC-Datei`*

Die folgenden Kommandozeilenoptionen werden von `abc2ly` unterstützt:

`-b,--beams=None`
: Die Balkensetzung aus der ABC-Datei erhalten.

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-o,--output=`*`Dateiname`*
: Die Ausgabe wird in die Datei *Dateiname*'`.ly`' geschrieben.

`-s,--strict`
: Strenge Auslegung einer erfolgreichen Konvertierung.

`-v,--version`
: Gibt die Versionsnummer aus.

Es existiert außerdem die Möglichkeit, LilyPond-Befehle für die Konvertierung mit `abc2ly` gleich in der ABC-Datei anzugeben. Wenn sich etwa in der ABC-Datei das Kommentar

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

befindet, so wird der Text nach dem Schlüsselwort ‚voices' direkt in die aktuelle Stimme in der LilyPond-Datei eingefügt.

Ebenso bewirkt

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

dass alles nach dem ‚slyrics' Schlüsselwort an der aktuellen Stelle im Liedtext eingefügt wird. Gleichermaßen wird mit

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

der Text, der auf das „slylrics"-Schlüsselwort folgt, in die aktuelle Gesangstextzeile eingefügt.

Bekannte Probleme und Warnungen

Der ABC-Standard ist eigentlich kein wirklich vollständiger Standard. Für komplexere Notation wie etwa Polyphonie existieren verschiedene Konventionen.

Mehrere Lieder in einer Datei können nicht konvertiert werden.

ABC synchronisiert den Liedtext am Anfang jeder Zeile mit den Noten, `abc2ly` macht dies nicht.

`abc2ly` ignoriert die Balkensetzung in der ABC-Datei.

4.3.4 `etf2ly` aufrufen

> **Achtung:** Dieses Programm ist nicht unterstützt und kann in kommenden LilyPond-Versionen entfernt werden.

ETF (Enigma Transport Format) ist ein Dateiformat, das Coda Music Technology in älteren Versionen des Programms Finale benutzt hat.

`etf2ly` konvertiert Teile einer ETF-Datei nach LilyPond und wird von der Kommandozeile folgendermaßen aufgerufen:

```
etf2ly [Optionen]... ETF-Datei
```

Unter ‚Kommandozeile' verstehen wir dabei die Kommandozeile des jeweiligen Betriebssystems. Für nähere Informationen hierzu siehe Abschnitt 4.3 [Von anderen Formaten konvertieren], Seite 35.

Die folgenden Kommandozeilenoptionen werden von `etf2ly` unterstützt:

`-h,--help`
: Zeigt eine Zusammenfassung der Programmbenutzung und der Optionen.

`-o,--output=`*Dateiname*
: Die Ausgabe wird in die Datei *Dateiname*‘`.ly`’ geschrieben.

`--version`
: Gibt die Versionsnummer aus.

Bekannte Probleme und Warnungen

Die Liste der Artikulationszeichen is unvollständig. Leere Takte verwirren `etf2ly`. Mehrfache Vorschlagnoten werden falsch beendet.

4.3.5 Andere Formate

LilyPond kommt nicht mit der Unterstützung für andere Formate, aber einige externe Programme können auch LilyPond-Dateien erstellen. Diese finden sich unter Abschnitt “Leichteres Editieren” in *Allgemeine Information*.

4.4 LilyPond-Ausgabe in anderen Programmen

Dieser Abschnitt stellt Methoden vor, wie Text und Musik auf andere Weise kombiniert werden können als dies durch `lilypond-book` automatisiert geschieht.

4.4.1 Viele Zitate aus einer langen Partitur

Wenn aus einer großen Partitur viele kleine Fragmente eingefügt werden sollen, kann dazu das ‚clip systems' Feature benutzt werden. Siehe Abschnitt “Notationsfragmente extrahieren” in *Notationsreferenz*.

4.4.2 LilyPond-Noten in OpenOffice.org integrieren

Musik im LilyPond-Format kann in OpenOffice.org eingefügt werden mittels OOoLilyPond.

4.4.3 LilyPond-Noten in andere Programme integrieren

Um die Ausgabe von LilyPond in anderen Programmen einzufügen, sollte `lilypond` anstelle von `lilypond-book` benutzt werden. Jedes Beispiel muss getrennt manuell erzeugt und ins Dokument eingefügt werden; für letzteres schlagen Sie bitte im Handbuch Ihrer Textverarbeitungs-Software nach. Die meisten Programme unterstützen das Einfügen von Grafiken im '`PNG`'-, '`EPS`'- oder '`PDF`'-Format.

Um den leeren Rand um die Notenzeilen zu verringern, können folgende Einstellungen benutzt werden:

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

{ c1 }
```

Eine '`EPS`'-Datei kann mit folgendem Befehl erzeugt werden:

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

eine '`PNG`'-Datei mittels:

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

4.5 Unabhängige `include`-Abschnitte

Einige Leute haben große (und nützliche!) Code-Abschnitte geshrieben, die man in vielen Projekten verwenden kann. Dieser Code wird möglicherweise auch eines Tages in LilyPond selbst integriert werden, aber bis dahin müssen Sie ihn selber herunterladen und mit `\include` selber einfügen.

4.5.1 MIDI-Artikulation

LilyPond kann benutzt werden, um MIDI-Ausgabe zu erzeugen, etwa um hörend zu korrigieren, was man notiert hat. Jedoch werden nur Dynamik, explizite Tempoänderungen und die Noten und Dauern selber in der MIDI-Datei aufgenommen.

Das *articulate*-Projekt ist ein Versuch, mehr Information in eine MIDI-Datei zu kopieren. Es funktioniert, indem man Noten, die sich nicht unter Bögen befinden, verkürzt, um die Noten zu „artikulieren“. Die Stärke der Kürzung hängt von den Aritkulationszeichen ab, die an die Noten gehängt werden: Staccato halbiert den Notwert, Tenuto gibt der Note ihre gesamte Dauer usw. Das Skript kann auch Triller und Doppelschläge in MIDI umwandeln, und man könnte auch andere Ornamente implementieren.

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

Bekannte Probleme und Warnungen

Die größte Einschränkung ist, dass man nur die Sachen beeinflussen kann, die man weiß: alles, was nur Textbeschriftung ist (anstelle einer Noteneigenschaft) wird immernoch ignoriert.

5 Vorschläge zum Schreiben von LilyPond-Eingabe-Dateien

Jetzt sind Sie so weit, größere Stücke mit LilyPond zu schreiben – nicht nur die kleinen Beispiele aus der Übung, sondern ganze Stücke. Aber wie geht man das am besten an?

Solange LilyPond Ihre Dateien versteht und die Noten so setzt, wie Sie das wollen, spielt es eigentlich keine Rolle, wie Ihre Dateien aussehen. Es gibt aber trotzdem ein paar Dinge, die man beim Schreiben von LilyPond-Code berücksichtigen sollte.

- Was ist, wenn Sie einen Fehler machen? Die Struktur einer LilyPond-Datei kann es erleichtern (oder erschweren), bestimmte Fehler zu finden.
- Was ist, wenn Sie Ihre Dateien mit jemandem austauschen wollen? Oder Ihre Dateien nach einige Jahren noch einmal überarbeiten wollen? Manche LilyPond-Dateien versteht man auf den ersten Blick, über anderen muss man eine Stunde grübeln, um die Struktur zu ahnen.
- Was ist, wenn sie Ihre Dateien auf eine neuere LilyPond-Version aktualisieren wollen? Die Syntax der Eingabesprache verändert sich allmählich mit Verbesserungen im Programm. Die meisten Veränderungen können automatisch durch `convert-ly` gelöst werden, aber bestimmte Änderungen brauchen Handarbeit. LilyPond-Dateien können strukturiert werden, damit sie einfacher aktualisierbar sind.

5.1 Allgemeine Vorschläge

Hier einige Vorschläge, wie Sie Probleme vermeiden oder lösen können:

- **Schreiben Sie immer mit `\version` die Versionsnummer in jede Datei**. Beachten Sie, dass in allen Vorlagen die Versionsnummer `\version "2.12.0"` eingetragen ist. Es empfiehlt sich, in alle Dateien, unabhängig von ihrer Größe, den `\version`-Befehl einzufügen. Persönliche Erfahrung hat gezeigt, dass es ziemlich frustrierend sein kann zu erinnern, welche Programmversion man etwa vor einem Jahr verwendet hat. Auch `convert-ly` benötigt die Versionsnummer.
- **Benutzen Sie Überprüfungen**: Abschnitt “Oktavenüberprüfung” in *Notationsreferenz*, und Abschnitt “Takt- und Taktzahlüberprüfung” in *Notationsreferenz*. Wenn Sie hier und da diese Überprüfungen einfügen, finden Sie einen möglichen Fehler weit schneller. Wie oft aber ist „hier und da“? Das hängt von der Komplexität der Musik ab. ei einfachen Stücken reicht es vielleicht ein- oder zweimal, in sehr komplexer Musik sollte man sie vielleicht in jeden Takt einfügen.
- **Ein Takt pro Textzeile**. Wenn irgendetwas kompliziertes vorkommt, entweder in der Musik selber oder in der Anpassung der Ausgabe, empfiehlt es sich oft, nur einen Takt pro Zeile zu schreiben. Bildschirmplatz zu sparen, indem Sie acht Takte in eine Zeile zwängen, hilft nicht weiter, wenn Sie ihre Datei „debuggen“ müssen.
- **Kommentieren Sie ihre Dateien**. Benutzen Sie entweder Taktnummern (in regelmäßigen Abständen) oder Verweise auf musikalische Themen („Zweites Thema in den Geigen“, „vierte Variation“ usw.). Sie brauchen diese Kommentare vielleicht noch nicht, wenn Sie das Stück notieren, aber spätestens wenn Sie nach ein paar Jahren etwas verändern wollen oder Sie den Quelltext an einen Freund weitergeben wollen, ist es weitaus komplizierter, die Dateistruktur ohne Kommentare zu verstehen, als wenn Sie sie rechtzeitig eingefügt hätten.
- **Schreiben Sie Klammern mit Einrückung**. Viele Probleme entstehen durch ungerade Anzahl von `{` und `}`-Klammern.
- **Schreiben Sie Tondauerangaben** am Anfang von Abschnitten und Bezeichnern. Wenn Sie beispielsweise `c4 d e` am Anfang eines Abschnittes schreiben, ersparen Sie sich viele Probleme, wenn Sie ihre Musik eines Tages umarrangieren wollen.
- **Trennen Sie Einstellungen** von den eigentlichen Noten. Siehe auch Abschnitt “Tipparbeit durch Variablen und Funktionen ersparen” in *Handbuch zum Lernen* und Abschnitt “Globale Formatierung” in *Handbuch zum Lernen*.

5.2 Das Kopieren von bereits vorhandener Musik

Wenn Sie Musik aus einer fertigen Partitur kopieren (z. B. die LilyPond-Eingabe einer gedruckten Partitur):

- Schreiben Sie ein System ihrer Quelle nach dem anderen (aber trotzdem nur einen Takt pro Textzeile) und überprüfen Sie jedes System, nachdem Sie es fertig kopiert haben. Mit dem `showLastLength`- oder `showFirstLenght`-Befehl können Sie den Übersetzungsprozess beschleunigen. Siehe auch Abschnitt "Korrigierte Musik überspringen" in *Notationsreferenz*.
- Definieren Sie `mBreak = { \break }` und schreiben Sie `\mBreak` in der Quelldatei immer dann, wenn im Manuskript ein Zeilenumbruch vorkommt. Das macht es einfacher, die gesetzte Zeile mit den ursprünglichen Noten zu vergleichen. Wenn Sie die Partitur fertig gestellt haben, könne Sie `mBreak = { }`, also leer definieren, um diese manuellen Zeilenumbrüche zu entfernen. Damit kann dann LilyPond selber entscheiden, wohin es passende Zeilenumbrüche platziert.
- Wenn Sie eine Stimme für ein transponierendes Instrument als eine Variable notieren, wird empfohlen, dass die Noten von

  ```
  \transpose c klingende-Tonhöhe {...}
  ```

  eingefasst werden (wobei `klingende-Tonhöhe` die klingende Tonhöhe des Instruments ist), sodass die Noten innerhalb der Variable für klingendes C geschrieben sind. Sie können die Variable zurücktransponieren, wenn es nötig ist, aber Sie müssen es nicht tun. Fehler in Transpositionen sind treten seltener auf, wenn alle Noten in den Variablen für die gleiche Ausgangstonhöhe geschrieben werden.

  Denken Sie auch daran, dass Sie nur von/nach C transponieren. Das heißt, dass die einzigen anderen Tonhöhen, die Sie in Transpositionen benutzen, die Tonhöhen der Instrumente sind, für die Sie schreiben: `bes` für eine B-Trompete oder `aes` für eine As-Klarinette usw.

5.3 Große Projekte

Besonders wenn Sie an größeren Projekten arbeiten, ist es unumgänglich, dass Sie ihre LilyPond-Dateien klar strukturieren.

- **Verwenden Sie Variablen für jede Stimme**, innerhalb der Definition sollte so wenig Struktur wie möglich sein. Die Struktur des `\score`-Abschnittes verändert sich am ehesten, während die `violine`-Definition sich wahrscheinlich mit einer neuen Programmversion nicht verändern wird.

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

- **Trennen Sie Einstellungen von den Noten**. Diese Empfehlung wurde schon früher gegeben, aber für große Projekte ist es unumgänglich. Muss z. B. die Definition für `fdannp` verändert werden, so braucht man es nur einmal vorzunehmen und die Noten in der Geigenstimme, `violin`, bleiben unberührt.

  ```
  fdannp = _\markup{
  ```

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

5.4 Fehlersuche

Früher oder später werden Sie in die Lage kommen, dass LilyPond Ihre Datei nicht kompilieren will. Die Information, die LilyPond während der Übersetzung gibt, können Ihnen helfen, den Fehler zu finden, aber in vielen Fällen müssen Sie nach der Fehlerquelle auf die Suche gehen.

Die besten Hilfsmittel sind in diesem Fall das Zeilen- und Blockkommentar (angezeigt durch `%` bzw. `%{ ... %}`). Wenn Sie nicht bestimmen können, wo sich das Problem befindet, beginnen Sie damit, große Teile des Quelltextes auszukommentieren. Nachdem Sie einen Teil auskommentiert haben, versuchen Sie, die Datei erneut zu übersetzen. Wenn es jetzt funktioniert, muss sich das Problem innerhalb der Kommentare befinden. Wenn es nicht funktioniert, müssen Sie weitere Teile auskommentieren bis sie eine Version haben, die funktioniert.

In Extremfällen bleibt nur noch solch ein Beispiel übrig:

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

(also eine Datei ohne Noten).

Geben Sie nicht auf, wenn das vorkommen sollte. Nehmen Sie das Kommentarzeichen von einem Teil wieder weg, sagen wir der Bassstimme, und schauen Sie, ob es funktioniert. Wenn nicht, dann kommentieren Sie die gesamte Bassstimme aus, aber nicht den `\bass`-Befehl in dem `\score`-Abschnitt:

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

Jetzt beginnen Sie damit, langsam Stück für Stück der Bassstimme wieder hineinzunehmen, bis Sie die problematische Zeile finden.

Eine andere nützliche Technik zur Problemlösung ist es, Abschnitt “Minimalbeispiele” in *Allgemeine Information* zu konstruieren.

5.5 Make und Makefiles

Fast alle Betriebssysteme, auf denen LilyPond benutzt werden kann, unterstützen ein Programm mit dem Namen `make`. Dieses Programm liest eine besondere Datei mit der Bezeichnung `Makefile`, die definiert, welche Dateien von welchen anderen Dateien abhängen und welche Befehle für das Betriebssystem nötig sind, um eine Datei aus einer anderen zu erstellen. Ein Makefile könnte etwa erklären, wie `ballad.pdf` und `ballad.midi` aus `ballad.ly` erstellt werden können, indem LilyPond aufgerufen wird.

Es gibt Fällen, wenn es sich sehr stark empfiehlt, ein `Makefile` für das aktuelle Projekt zu erstellen, entweder zur eigenen Bequemlichkeit, oder aber auch als Hilfe für andere, die vielle-

icht einmal die Quelldateien lesen und verstehen wollen. Insbesondere bei großen Projekten mit vielen eingefügten Dateien und unterschiedlichen Ausgabeoptionen (etwa Partitur, einzelne Stimmen, Dirigierpartitur, Klavierauszug usw.), aber auch bei Projekten, die komplizierte Programmaufrufe zur Verarbeitung erfordern (wenn man etwa mit `lilypond-book` arbeitet), lohnt sich die Erstellung einer Make-Datei. Diese Dateien können sehr unterschiedliche ausfallen, und ihre Komplexität und Flexibilität kann den Bedürfnissen aber auch Kenntnissen des Schreibers angepasst werden. Das Programm GNU Make ist auf GNU/Linux Distributionen und MacOS X installiert, aber es ist auch für Windows erhältlich.

Das **GNU Make Manual** gibt eine vollständige Anleitung, wie `make` benutzt werden kann. Hier sollen nur einige kleine Blicke auf die vielfältigen Möglichkeiten geworfen werden.

Die Befehle, um Regeln in einer Make-Datei zu erstellen, unterscheidet sich zwischen den Betriebssystemen. Die verschiedenen Linuxe und MacOS X benutzen `bash`, während unter Windows `cmd` eingesetzt wird. Unter MacOS X muss man das System so konfigurieren, dass die Kommandozeile benutzt wird. Hier einige Beispiele für Make-Dateien, mit Versionen für Linux/MacOS und Windows.

Das erste Beispiel ist für ein Orchesterstück in vier Stätzen unt mit der folgenden Dateistruktur:

```
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
```

Die `.ly`-Dateien un den Verzeichnissen `Scores` und `Parts` erhalten ihrere Noten aus `.ily`-Dateien, die sich im `Notes`-Verzeichnis befinden:

```
%%% Kopfzeile der Datei "symphony-cello.ly"
\include ../definitions.ily
\include ../Notes/cello.ily
```

Die Make-Datei hat die Ziele `score` (das gesamte Stück als große Partitur), `movements` (die einzelnen Sätze als große Partitur) und `parts` (die einzelnen Stimmen für die Spieler). Es gibt auch das Ziel `archive`, welches ein Tar-Archiv der Quelldateien erstellt, etwa um die Quellen über das Internet oder per E-Mail zu verteilen. Hier die Make-Datei für GNU/Linux oder MacOS X. Sie sollte unter dem Namen `Makefile` im obersten Verzeichnis des Projektes gespeichert werden:

> **Achtung:** Wenn ein Ziel oder eine Musterregel definiert ist, müssen die folgenden Zeilen mit Tabulatoren, nicht mit Leerzeichen beginnen.

```
# Namensstamm der Ausgabedateien
piece = symphony
# finde heraus, wieviele Prozessoren vorhanden sind
CPU_CORES=`cat /proc/cpuinfo | grep -m1 "cpu cores" | sed s/".*: "//`
# Der Befehl, um lilypond aufzurufen
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click -djob-count=$(CPU_CORES)

# Die Endungen, die im Makefile benutzt werden
.SUFFIXES: .ly .ily .pdf .midi

# Eingabe- und Ausgabedateien werden in den Verzeichnissen durchsucht,
# die sich in der VPATH-Variable befinden.  Alle sind Unterverzeichnisse
# des aktuellen Verzeichnisses (angegeben durch die GNU make-Variable
# `CURDIR').
VPATH = \
  $(CURDIR)/Scores \
  $(CURDIR)/PDF \
  $(CURDIR)/Parts \
  $(CURDIR)/Notes

# Die Musterregel, um PDF und MIDI-Dateien aus der LY-Eingabedatei
# zu erstellen.  Die .pdf-Ausgabedateien werden in das
# `PDF'-Unterverzeichnis abgelegt, die .midi-Dateien in das
# `MIDI'-Unterverzeichnis.
%.pdf %.midi: %.ly
        $(LILY_CMD) $<; \           # this line begins with a tab
        if test -f "$*.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
```

```
# Abhängigkeiten der einzelnen Sätze.
$(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)

# Abhängigkeiten der großen Partitur.
$(piece).pdf: $(piece).ly $(notes)

# Abhängigkeiten der Stimmen.
$(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

# `make score' eintippen, um die große Partitur mit allen vier
# Sätzen als eine Datei zu erstellen.
.PHONY: score
score: $(piece).pdf

# `make parts' tippen, um alle Stimmen zu erstellen.
# `make foo.pdf' tippen, um die Stimme für das Instrument `foo' zu erstellen.
# Beispiel: `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

# `make movements' tippen um Dateien für die vier Sätze einzeln zu erstellen.
.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/*
```

Unter Windows ergeben sich bestimmte Komplikationen. Nachdem man GNU Make für Windows heruntergeladen und installiert hat, muss man den richtigen Pfad in den Umgebungsvariablen des Systems setzen, damit die DOS-Kommandozeile das Make-Programm finden kann. Um das vorzunehmen, kann mit der rechten Maustaste auf "Arbeitsplatz" klicken, dann

`Eigenschaften` und `Erweitert` geklickt werden. Hier wählt man `Umgebungsvariablen`. In der Liste `Systemvariablen` wählt man `Path` und mit einem Klick auf `Bearbeiten` kann man den Pfad zu der `.exe`-Datei von GNU Make hinzufügen, der etwa wie folgt aussieht:

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

Die Make-Datei selber muss auch angepasst werden, um unterschiedliche Shell-Befehle zu verwenden und mit Leerzeichen umgehen zu können, die sich in einigen Standardverzeichnissen unter Windows befinden. Das `archive`-Ziel wird entfernt, da Windows den `tar`-Befehl nicht kennt, und Windows benutzt auch eine andere Dateiendung für midi-Dateien.

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

# 8.3 Bezeichnung für CURDIR erhalten (Workaround wg. Leerzeichen 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) $<      # diese Zeile beginnt mit Tabulator
        if exist "$*.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
```

Die nächste Make-Datei ist für ein `lilypond-book`-Dokument, das in LaTeX gesetzt wird. Das Projekt hat einen Index, welcher erfordert, dass der Befehl `latex` zweimal aufgerufen wird, um die Verweise zu aktualisieren. Ausgabedateien werden in einem `out`-Verzeichnis für die .pdf-Dateien gespeichert und in `htmlout` für die html-Dateien.

```
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: soll auch unter Windows funktionieren

Die vorige Make-Datei funktioniert nicht unter Windows. Als Alternative für Windows-Benutzer könnte man eine einfache batch-Datei erstellen, welche die erforderlichen Befehl enthält. Sie kümmert sich nicht um Abhängigkeiten, wie es eine Make-Datei kann, aber wenigstens wird die Kompilation auf einen einzigen Befehl beschränkt. Das folgende kann als Datei `build.bat` oder `build.cmd` gespeichert werden. Die Batch-Datei kann auf der Kommandozeile aufgerufen werden oder einfach doppelt angeklickt werden.

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

Siehe auch

Programmbenutzung: Abschnitt “Benutzung auf der Kommandozeile” in *Anwendungsbenutzung*, Abschnitt “lilypond-book” in *Anwendungsbenutzung*.

Anhang A GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://fsf.org/

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

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

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

4. MODIFICATIONS

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

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

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

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

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

Anhang B LilyPond-Index

\

\header in LaTeX-Dokumenten ... 19

A

ABC ... 38
Aktualisierung von alten Eingabedateien ... 12
Aktualisierung von LilyPond-Datei ... 12
Aufruf von dvips ... 26
Ausgabedatei, Dateigröße ... 34
Ausgabedateiname ... 4
Ausgabeformat, Definition von ... 3

C

Coda Technology ... 39
convert-ly ... 12

D

Dateigröße, PDF ... 34
Dateisuche ... 4
docbook ... 16
DocBook, Noten in ... 16
Dokument, Noten einfügen in ... 16
dvips ... 26

E

Editoren ... 35
emacs ... 35
enigma ... 39
EPS (encapsulated PostScript) ... 3
error messages ... 8
ETF ... 39
Externe Programme, LilyPond-Dateien erstellen .. 39

F

Fataler Fehler ... 8
Fehler ... 8
Fehlermeldung, Format ... 8
Fehlermeldungen ... 8
Fehlerprotokoll, Scheme ... 8
Finale ... 39
Form der Fehlermeldungen ... 8

H

Hervorhebung, Syntax ... 35
Hilfe, Kommandozeile ... 2
html ... 16
HTML, Noten in ... 16

K

Kommandozeile, `lilypond` aufrufen ... 2
Kommandozeilen-Optionen für `lilypond` ... 1
Konturschriften ... 26

L

LANG ... 6
latex ... 16
LaTeX, Noten in ... 16
`lilypond` auf der Kommandozeile ... 2
`lilypond` aufrufen ... 1
LILYPOND_DATADIR ... 6

M

make ... 43
Make-Dateien ... 43
Makefile ... 43
`Manuals` ... 1
MIDI ... 36
Modi, Editor ... 35
MusicXML ... 37
Musikwissenschaft ... 16

O

OpenOffice.org ... 40
Optionen an der Kommandozeile ... 1
Optionen, Kommandozeile ... 2

P

Papierformat, Kommandozeile ... 2
PDF-Ausgabe ... 4
PNG-Ausgabe ... 4
point and click ... 34
Point and Click, Kommandozeile ... 2
Portable Document Format (PDF) ... 4
Portable Network Graphics (PNG) ... 4
PostScript Ausgabeformat ... 3
PostScript-Ausgabe ... 4
Programmierfehler ... 8

S

safe, Kommandozeile ... 2
Scheme dump ... 4
Scheme Fehler ... 8
Scheme-dump ... 4
sicher, Kommandozeile ... 2
Suchpfad ... 4
SVG (scalable vector graphics) ... 3
Syntax highlight ... 35
Syntaxhervorhebung ... 35

T

Terminal, `lilypond` aufrufen ... 2
texi ... 16
texinfo ... 16
Texinfo, Noten in ... 16
Titel und lilypond-book ... 19
Type1 Schriften ... 26

U

Umgebungsvariablen 6
Update von alten Eingabedateien 12

V

Variablen, Umgebungs- 6
Vektorgraphik (SVG) 3
Verzeichnis, Ausgabe speichern in 4
vim 35
Vorschau 4

W

Warnung 8