

LilyPond

AUTOMATED MUSIC ENGRAVING -2010年3月2日-

Tutorial

LilyPond language 譜面原稿記述において

LilyPond のデフォルトは **nederlands.ly**(In Dutch=オランダ語表記) を読み込みで、与えられたその原稿処理を行います。音楽における **Note Names**, **sharp**, **flat** 等々の strings には国(文化)間で差異がありますので的便に置き換えてください。Lilypond が提供しているそれらは、

```
\include "english.ly" , \include "deutsch.ly" , \include "norsk.ly" , \include "svenska.ly" ,  
\include "italiano.ly" , \include "catalan.ly" , \include "espanol.ly" , \include "arabic.ly" ,  
\include "portugues.ly" , \include "suomi.ly" , \include "vlaams.ly" 等々です。
```

これらは、譜面原稿ファイル一番最初に宣言(冒頭最初の行に記述)します—下記の如く—。

```
\include "englishe.ly"  
\version "2.12.1"  
#(set-default-paper-size "a4")
```

LilyPond の解説書はそのバージョンによる、オンラインファイルである、LilyPond.pdf(印刷本)・LilyPond documentation(WEB ファイル) 等々を参照してください。

<http://lilypond.org/doc/v2.12/Documentation>

—譜面知識さえあれば、(英語文意に頼らずとも)直感的に理解出来る様になっています。—

LilyPond のプログラムの興味は、例えば、

C:...\\lilypond\\...\\ly\\engraver-init.ly 等々の内容記述に目を通すことを進めます。

文章の中の、” \ (エスケープ)”は日本語キーボードでは半角¥あるいは、半角\で打ち込むことを意味しています。

LilyPond での譜面コントロール記述は、何よりも、まず譜面知識を持つ事の優先を推奨します。その知識で直線的に目線位置(英語解説文章に左右されず)で LilyPond 作用は理解できます。更には、音楽を作る(作曲等々)等々の力も備わります。LilyPond のコマンド知識のみで音楽等々は作曲できません。

原稿を音符コーディングしている時は、**日本語特有の全角空白、全角文字は使用禁止(lyrics-歌詞-を除く)です**。LilyPond の lyricText は utf-8 code で記述(組版には各国のフォントが必要)です。

デフォルト組版は 四分音符・4/4 拍子・ト音記号・ハ長調で組版出力されます。mid は指定楽器がなければ、"acoustic grand" (ピアノ) で作成されます。

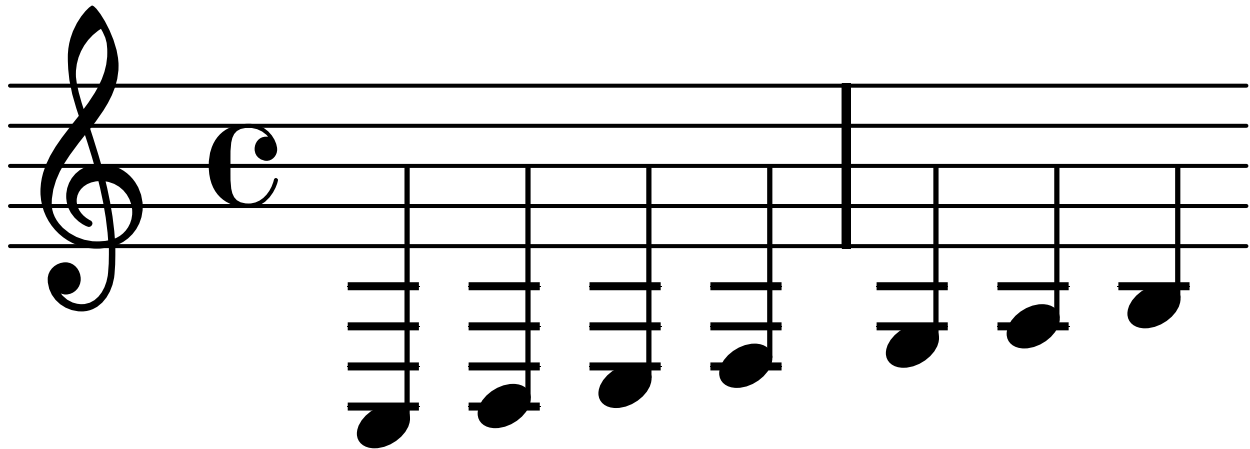
First steps

● 始めの一步

例えば、何かまわず以下のようにテキストファイルへ typing して、LilyPond でコンパイルすると。

```
{ c d e f g a b }
```

結果の表示です (A4 版で組版されます, 此处では obje 部分だけを切り抜いて表示).



- テキストファイルは Windows OS の「メモ帳」等々を使用して作成できます.
- ファイルネームは任意ネームで, 例えば `exa-a.ly` 等々と名前をつけます (和語名は駄目です, 半角アルファベット `a.....z` の 26 文字です).

ここで注意は拡張子を必ず `.ly` にします.

- コンパイルは, Windows OS のコマンドプロンプト (`cmd.exe` 通称 dos 窓) を開いて

(`exa-a.ly` のある場所で)

>lilypond `exa-a.ly` と入力し [Enter] を押せば OK です.

色々表示しますが (文字化けは無視してください (これは正常な証です)).

Windows OS(のコマンドプロンプト-`cmd.exe`-) が和語表示は SIFT-JIS コード標準だからで, `lilypond.exe` の (メッセージ) 言語 (出力) 表示は utf-8 コードです

最終的に, 例ですと...`exa-a.ps` `exa-a.pdf` (命令 `\midi{}`) が有れば `mid` 等々が作成されています.

- 拡張子 `.ps` `.pdf` ファイルが譜面ファイルで, アクロバット等々で表示印刷出来ます.
- 拡張子 `.mid` が, 通称 `midi` ファイルと呼ばれてファイルで, Windows メディアプレーヤーで音 (音楽) が鳴ります.

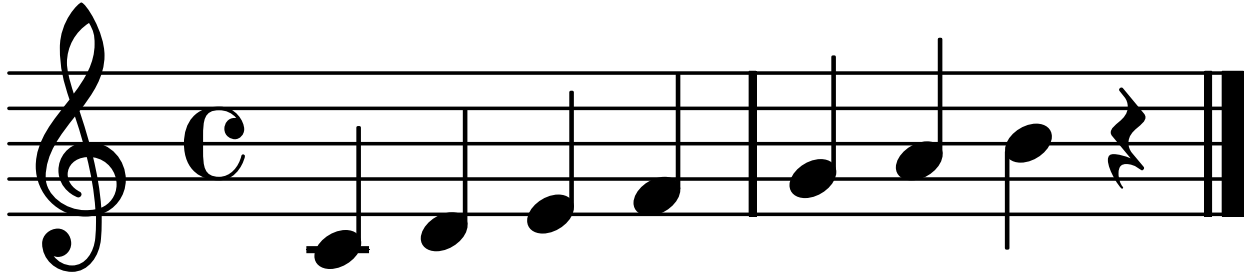
(Windows OS の MIDI 音源モジュールは, ライセンス版;Roland 社 (米国)GM/GS(R) です)

LilyPond が作成した `mid` ファイルは MIDI 規格標準の `format1 type` で, MIDI interface が備え付けられている電子機器で, これら `mid(midi)` ファイルをその音楽編集装置に取り組む事は (DAW ソフトで) 可能です. 但し, パソコン内の設定や音楽編集ソフトの設定および `midi` とオーディオ (Audio) 関連に, 相当に詳細な知識と技術を持つ必要があります.

- もう少し整理 (清書) しますと.....

```
{ \clef treble \key c\major \time 4/4
c'4 d' e' f' | g' a' b' r \bar "|." }
```

結果の表示です (obje 部分を切り抜いて表示).



- { } で囲む事をグルーピングと言いますが, 通常はここから.....此処までと cpu に計算の範囲を知らせます. このグルーピングは{ { } { } }等と多くのこの手のソフトプログラミングはネスティング (ネスト構造) にも対応しています.

- \文字列 の\は一般的にエスケープ文字と通称して, \文字列 で明示的に機能 (コマンド) を意明しています. \clef の和語訳は"音部記号"の意味で treble はその記号種の中の一つにすぎません, 他に bass 等々 LilyPond は多くをサポートしてあります, それらは全て \key "調号" と密接に繋がって \clef も \key も \Staff や \Voice に絡んでいます. つまり, \Staff や \Voice が無い所ではまったく何も作用もしません. プログラマーは良く \clef の value(値) は treble 等々と言います. 同様に \time の値は 4/4 等となりますが, どちらも値は文字列です. LilyPond language に於いて数字を cpu に数値と認識させるには接頭語 # を付けて, 例えば \hspace #3.5 とか \hspace #-2 等と記述します.

- プログラマーは, また, 広義の意味でならキーボードからの打ち込み (入力) は全てコマンドなのですと指摘します. そこまで行かなくとも, 上記 2 例 (楽譜) の LilyPond のコード記述は全てコマンドを意味している事 (文字出力でなく) が分かると思います. LilyPond での文字列 (譜面表示のテキスト) の入力は通常 \markup{...テキスト...} と括って記述します, \Lyrics(歌詞パート) の中での記述は, もちろん歌詞の文字列と認識されます.

上記これら 2 例は単なる lilyPond 単純理解表示の為のものです.

つまり, このように記述して. ファイル名前を **任意名.ly** で作成し, cmd.exe(Windows OS コンソール画面, プロンプト) >lilypond 任意名.ly[Enter] で楽譜やその mid(曲) を作成できます.

従いまして, **lilyPond が解釈できるシーケンス (処理命令文字列), コマンド (処理命令), lilyPond の決まりごと**に則って,

著者は原稿ファイルを記述しなければなりません. 上記 2 例はその一部分で LilyPond のデフォルト処理の結果を示しています.

注意が必要なのは, 始めて LilyPond コード記述の体験ならキーボードからの入力ミスは必ず付き物です. 慣れてくれば入力ミスの頻度は殆ど無くなる様になります.

音(階)は **c d e f g a b** のみです, 半音の上下は **is** か **es** を添え付けて **ais** や **aes** と記述し, 音(符)は **1 2 4 8 16 32 64** と数字を添え付けて例えば **a1** や **aes8** 等々と記述をします. また, **Octave** 上がる毎に'を追加し, **Octave** 下降する度に, を追加し例えば **aes"4** とか **a,2** 等々と記述する. 基本的にはたったこれだけです, これらの展開に `\clef` や `\major` や `\key` や `\time` や `\tempo` 等々があり, これらが基本的には音楽製作現場で飛び交い, あるいは, 曲作成原稿ファイルに書き込みます. LilyPond の基本的な音楽書き込みはこれだけです. 上記 2 例の如く書き込みすれば, その楽譜も `mid`(曲) も作成してくれます.

(**r**, **R** 音符は休符を意味し, **s** 音符は LilyPond 独特の幽霊音符を表します)

LilyPond のデフォルト処理とは, 著者が何も指定していなければ, その部分は LilyPond(プログラム) 自身の設定に沿って補完をして, `runing` 処理をする意味です.

`midi` の作成に関しては, `\midi{ }` の指定が書かれていない限り作成はしません.

- 1). 本格的な清書記述の一例です.....

```

\version "2.12.1"
exa = {
  c'4 d' e' f' | g' a' b' r \bar "|"
  \break
}
\score { \new Staff {
  \clef treble \key c\major \time 4/4 \tempo 4 = 80
  \exa
}

\layout {}
\midi {}
}

```

結果の表示です (obje 部分を切り抜いて表示).



解説

¶ `\version "2.12.1"`

これは、"譜面原稿ファイルは LilyPond version 2.12. 版で記述しました" の意味です。LilyPond は絶えず美的楽譜追求へ変貌し続けています (演奏者がおもわず弾きたくなるような譜面組版)。従って...version によって lilypond 自身の内部処理に差異が激しく生じます。譜面原稿ファイルへ version 情報記述は極めて重要です。

(ver2.12. であるなら、それ以後の数字はマイナーレベルなので、ver2.12.2 でも、ver2.12.12-4 等々に進化しても OK です)

マイナーレベルとは、ある種の環境 (特殊環境) による不都合を修正した版で、ver2.12 のメジャー版にはまったく影響を及ぼさないプログラム記述修正版です)

¶ `exa = { }`

exa という名前付けは任意名前で OK です。

この記述方式 (方法は) はマクロ記述フォームです。譜面コード記述を書き終えたら必ず } で閉じます, この閉じコード記述を忘れると lilypond コンパイルはその時点で解釈不能...中止終了します。

¶ `c'4 d' e' f' | g' a' b' r \bar "|" .`

`\break`

`\clef treble \key c\major \time 4/4 \tempo 4 = 80`

具体的な、これらのコーディング文字意味等々は [http://lilypond.org/doc/v2.12/Documentation/Notation Reference \(in one big page ~ 4 MB, in PDF\)-LilyPond.pdf](http://lilypond.org/doc/v2.12/Documentation/Notation%20Reference%20(in%20one%20big%20page%20~%204%20MB,%20in%20PDF)-LilyPond.pdf) 等々を参照してください。内容は譜面知識さえお持ちなら理解できます (英語は関係ない)。

¶ `\score { ... }`

`\score {` は lilypond の楽譜や譜面処理の開始命令です。音楽原稿ファイル进行处理すべき様々な内部ファイルが読み込まれ、著者の原稿进行处理できるように用意されます。これも従いまして、`}` で閉じ—此処で楽譜や譜面の処理範囲終了明示—がなければなりません。忘れると lilypond は処理を中止します。

(正確に言及すると、最初に `\book {.....` の宣言があり、その原稿ファイルの一番最終行で `.....` } の綴りで全てが終了です、これは省略でき多くの LilyPond 紹介では省略して行っています)

¶ `\new Staff { ... }`

読み込んだ著者の楽曲原稿ファイルを、音楽の五線譜上で組版処理しなさいです。明確のように `}` でその範囲終了を忘れないでください。

¶ `\exa`

`exa` 名前前で記述した音楽コーディングをこの場所に呼び込んで処理してくださいの意味です。

この `exa` 名前付けは任意命名です。呼び出すときは LilyPond エスケープ文字 `"\"` を接頭語に付けます。

¶ `\layout { }`

これは、処理音楽情報 (組版処理) を楽譜 (譜面) としたファイル作成せよの命令です。

`default` は音楽界 A4 版譜面用紙になります (文書用 A4 版より約 10.00mmm 大きいです)

`{ }` の中に組版の際の様々なパラメータを指定できます。

更に、上記の `\new Staff` での呼び出す Staff への注文設定も、此処で記述できます

LilyPond プログラム内部構築 (architecture) 部分でなら、ここから、外部ソフト `gs(unix=Windows)` へ処理メモリーが渡され、`gs` ソフトによって `ps,eps,png,pdf` (楽譜) 処理へと移行します [注意: Winodws 版も `unix` 版も同じ LilyPond 内で `gs(unix 系用)` を用いています]。 `gs` とは Ghostscript の略語で PostScript (Adobe Systems compay 開発ライセンス) のページ記述言語の事で、Adobe 社の了解を得たフリーライセンス版です (Adobe 社の製品版は Acrobat)。 `gs` と言えども、法人でお使いの場合は使用書体ともども URL には注意が必要です。

¶ `\midi { }`

これは、処理音楽情報を `mid`(曲) ファイルとして作成せよの命令です。

`{ }` の中に `dynamics` 設定等々様々な曲に関係するパラメーターを記述できます。

`mid` データーは LilyPond では `C:\Program Files\LilyPond\usr\lib\lilypond\current\python2.4\` のパッケージファイル群によって裁かれます (専用 Windows OS 用へのファイル有り) が、`mid` ファイルフォーマットは多分に `type1` の世界標準になっています。従って、市販の音楽制作 `pc` ソフト (録音スタジオデスクトップ環境ソフト) に呼び込んでも `mid` 情報は再現します。つまり、イベントリストに仕分けられた LilyPond の `mid` 情報は参照できます。である限り、LilyPond の `mid` 情報を音楽制作 `pc` ソフト (録音スタジオデスクトップ環境ソフト) で再編集は可能です (LilyPond の `mid` へ更なる。音の幅や深さや曲感を求めるなら、`mid` 編集ソフト側に呼び込んで `mid` 情報を再制御)

LilyPond(言語記述)でどのような命令であれ、{ で記述を始まりだしたら必ず } でその範囲終了を確実に明示しなければなりません...これは lilypond(言語記述)の法則です。余談ですが LilyPond は music typesetter というオペレーション(OS)で Windows OS や UNIX(Linux) 等々と同じです。ただ、印刷や表示等々を独自に用意するのではなく Windows OS や UNIX 等々の API を利用する方法を採っているに過ぎません。

LilyPond が楽譜を組版処理するとき、従来からの音楽界の法則に則って処理をします。LilyPond が独自に決めて処理しているわけでは有りません。ですが、LilyPond の内部処理コード(プログラム言語)に長じているなら自己独自の音楽記号等々など楽譜ファイルへ組版可能です。LilyPond の楽譜部分のプログラム言語は"scheme"であり、且つ"python"で運転を行い。コンピューター CPU との会話は等々(shell)は"bash"(UNIX(Linux), cygwin) や"cmd.exe"(Windows OS) です。

`\new Staff {.....}`, `\new Voice {.....}` の代わりに、

`\context Staff = "任意命名 ID 名前" {...}` や

`\context Voice = "任意命名 ID 名前" {...}` のコーディングもあります。

これらは、より詳細でより複雑な階層的な音楽記述するときに利用します(組版結果が単純でも音楽記述が複雑な場合)。プログラムの言うところ、翻訳(計算)し終わった作用後(結果)を格納しているメモリーに名前(メモリー番地)を付けている行為です。

これは、名前をつけていれば.....何時でも何処にでも.....そのメモリー(結果)を譜面や紙面に呼び出す事が出来る、その名前の `Staff`, `Voice` へ後からいろいろと追加作業が出来るからです。

`\new Staff` や `\new Voice` は、名前が無いので其のまま次の `job-obje` へと時系列で流れ込みであり、振り返っても、これから起こるずっと先からでも、その結果を利用できません。一回限りの終了です。

個々のコマンドは夫々の `\Voice or \Staff or \score` のどれかに所属していますが、一番始めに(最底部分)Voice Context の所属するコマンドが翻訳を実行、それを Staff Context へ収検され、Score Context によって楽譜情報となり。それらが PostScript 言語に直訳され.....Ghostscript の `ps2pdf` 等々で譜面になり印刷・表示となる。

これが LilyPond の処理の流れですが、気を付けるのは、作用設定の変更は、其の所属する Context の中で行うです。

また、いきなり `\new Staff` 宣言で音符等々を書き始めているのは、Staff Context は Voice Context を受け取る.....事が出来る。この文言の一文があるので、多くの user は `\new Staff` 等々からはじめています。

従って、`\new Staff` で音楽を書き始めて `Voice` 所属のコマンドを `\override` で修正しようとも.....それは出来ません。

どのコマンドが何処に所属しているかは、

C:\Program Files\LilyPond\usr\share\lilypond\current\ly\engraver-init.ly を参照願います。

Second steps

• lilypond コードに於ける " s " の役割

LilyPond に於ける " s " 音符コードは非常に重要です. 一つのサンプルコード例です.

```
\version "2.12.1"
exa = {
  s1_\markup{ \large \box {s } is ghost! } | s2 s |
  s4^\< s s\sustainOn s\! | s2^\> s\! | s1^\fermata \bar "|."
  \break
}

\score { \context Staff = "example4" {
  \clef treble \key c\major \time 4/4 \tempo 4 = 80
  \exa
}

\layout {}
\midi {}
}
```

結果の表示です (obje 部分を切り抜いて表示).

" s " 音符コードで記述した音符は幽霊です! lilypond コンパイル上は run しますが, 組版結果には音符が幽霊で存在しませんし, 作成指示した mid へも作用しません.

しかし, 音楽処理 (パソコンの CPU 計算) 上では例の如く正常に計算されています. 姿が透明だけに色々様々に組版では利用価値があります.

注意; LilyPond 特有のこの発明 s 音符は, 各人のデスクトップ環境の"録音スタジオ編集ソフト"(音楽制作ソフト)には無い機能です. 現在 (2008 year) の代表的なこれら高価なソフト群では, 同じ mid 規格でも対応されていないはずです. mid を呼び込んだときに注意が必要です. LilyPond で音楽原稿ソース記述の時に楽譜譜面 out と mid(音楽曲)out をセパレートで記述してあり, mid-out 記述には s 音符の無い事を確認していれば関係ないですが.

● 譜面記述テクニック "s" の役割

下記の記述を見てください、譜面記述で.....悩んだ時"s"音符を思い出してください。
—更に、一気に LilyPond の音楽処理モード説明に入ります—

```
\version "2.12.1"
exa = {
  \clef bass \key e\major \time 2/4
  r8 b,8^\mp^\< fis[ cis']\! | % 1小節
  cis'8^\> b4.*1/3 s4.*1/3 s4.*1/3\! | % 2小節
  \break
  \clef treble \key c\major \time 4/4
  << { c''4^\ff c'' c'' c''\(\ } \ \ { e'2 e' } >> | % 3小節
  << { g'2\ s2 } \ \ { s2 f'\(\ } >> | % 4小節
  << { d''1 } \ \ { c'4\ d'8[ e' f'] a'4. } >> \bar "|" % 5小節
}

\score { \context Staff = "example5" {
  \tempo 8 = 40
  \override Score.BarNumber #'break-visibility = #all-visible
  \override Score.BarNumber #'break-visibility =
    #end-of-line-invisible
  \override Score.BarNumber #'stencil =
    #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \exa
}

\layout { }
\midi { }
}
```

結果の表示です (obje 部分を切り抜いて表示).

(mid 曲を検証してみてください. mp,ff,(de)crescendi が曲に反映されています)

2小節目と4小節目が"s"を使った組版です.

4小節目のコーディングに注意してください. ちなみに、ここを `g'2\ f'\(` と記述してコンパイルすると、Slur の処理 error("原稿記述の Slur の閉じ記述に誤り有り"意味) メッセージが画面に表示されますが、コンパイルは原稿譜面も mid も作成します. 比較すると一目瞭然で slur 部分 (弧線) が組版外になっています.

3小節目以後のコーディングに見られる譜面組版コーディングですが.

`<< {五線譜上位置} \ \ {五線譜下位置} >>` これが LilyPond の法則です. もちろん、"上位置\\下位置"の小節目数は一致していなければなりません.

—4小節目が単一に見えますが...実態は上下に音符を持った譜面です—

このサンプルが意味しているのは、

LilyPond に於いて << {五線譜上位置} \\ {五線譜下位置} >> でコーディングを始めた時に, "五線譜上位置で発信した属性は必ず五線譜上位置で受け止める" "五線譜下位置で発信した属性は必ず五線譜下位置で受け止める" これは鉄則です.

この意味でも, 幽霊音符 " s " (実態は有るけれど姿は透明な音符) 利用の一つの例です. プログラミングで語ると cpu 計算上は実態があって計算され, 型 (譜面/mid) を持つとき透明化してしまう obje という意味になります.

但し, いくら " s " 音符が透明でも実態は存在しているので音符のカウントはなされています. 例えば, \time 2/4 で 1 小節内に 4 分音符で 2 個以上の値の音符を " s " だからといって記述は出来ません. それが, 上例 2 小節目でのコーディング意味になっています.

• 和音的の組版



Polyphony のもう一つの和音的組版例です, 和音コード類似のコーディングの組版です. c d e f g a b を一気に 4 分音符と全音符で弾かしてみました. ソースコーディングは以下の簡単な実験的な記述です.

```
\score{ {<c' d' e' f' g' a' b'>4 r2. | <c' d' e' f' g' a' b'>1 | }
\layout { indent = 0.00\mm line-width = 60.00\mm } \midi{ } }
```

ご自分でテキストファイルへコーディングして, lilypond でコンパイルして mid を聴いてみてください. 上記のようなコーディングはゲーム感覚で実験的な遊び要素的な楽しみです. パソコンの前で.....いくらでも, ご自分の創意で楽しめます.

\layout { indent = 0.00\mm line-width = 60.00\mm } はこの文書での貼り付けの為に記述したのであって, 実際の実験では \layout{} で済みます.

ちなみに, Windows OS のワードプロセッサであるワードや, 非常に優れたソフト・エクセルへも, この文書の様に文章の中に譜面を貼り付けは可能です.

music typesetter である LilyPond は, 使い方によって **Music Game** ソフトとしてパソコン上で遊べます.

• LilyPond で遊ぶ楽しむ (音のゲーム)

この文書の最終的な目標は, この LilyPond でゲームを始めよう! が目的でした. まず第一に **LilyPond** コンパイル後に自動で譜面表示/mid 奏でを自動で行う, 簡単なバッチファイルの作成から始めます.

以下の内容を記述したファイルを作成します. ファイルネーム **ly-run.cmd**

(バッチファイルの名前は, 任意名.cmd が鉄則です (Windows OS 側の法規)).

```
lilypond %1.ly
%1.pdf | %1.mid
```

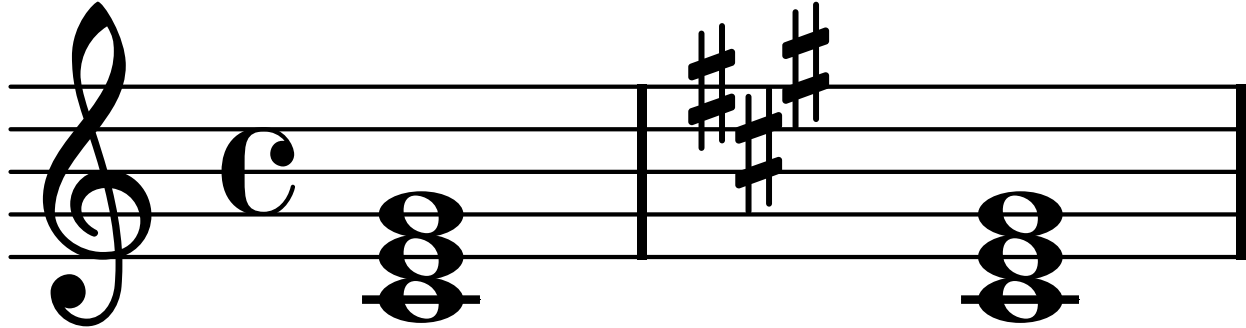
これは, lilypond の任意名前の原稿ファイルをコンパイル後に pdf ファイルと mid ファイルを自動 open する極めて簡単な Windows OS でのバッチファイルです.

(注意; Windows OS のバッチファイル (***.cmd or ***.bat) 内での記述の%は Windows OS のみ参照の引数機能で, LilyPond 等々他には関係ないです)

例えば, 以下の原稿記述でファイルを新規作成しまして, ファイルネーム **test.ly** (音楽原稿ファイル名前は, 任意名.ly が鉄則です (LilyPond OS 側の法規)).

```
\score{ {\key c\major <c' e' g'>1 | \key a\major <cis' e' gis'>1} \layout{} \midi{ } }
```

ファイル名を test.ly 等と名づけて保存し、
cmd.exe(コマンドプロンプト) で >ly-run test[Enter] とすれば、test.pdf と test.mid が自動で
open します。



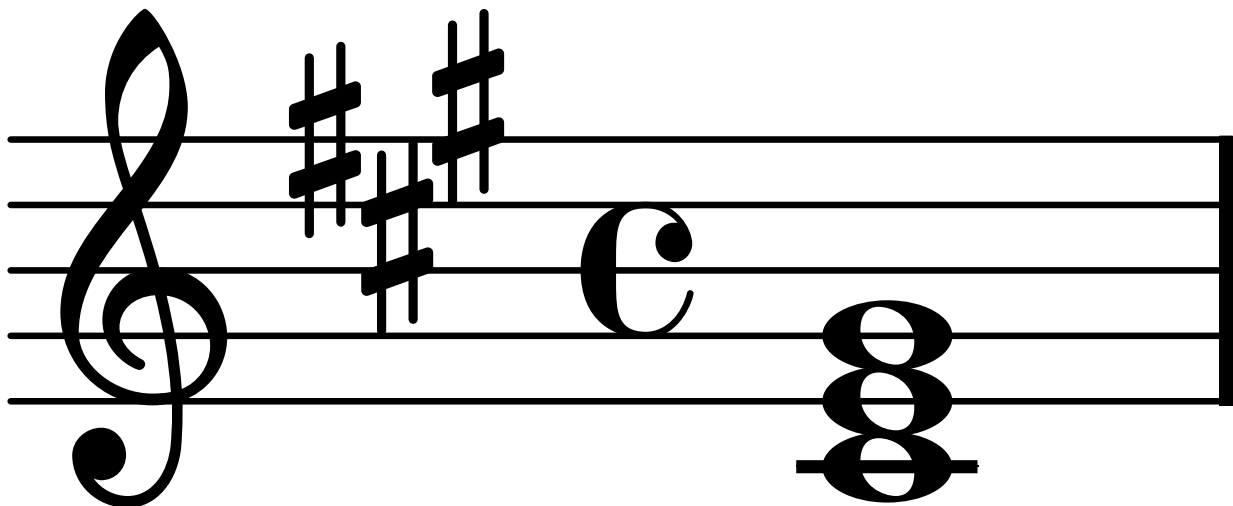
ここで、注意が必要なのは、ly-run.cmd も test.ly も同じ場所内 (フォルダー or ディレクトリ) に有ることです (一番簡単な方法)。その場所内で、cmd.exe(コマンドプロンプト) で >ly-run test[Enter] と実行すること、曲終了後は、wmplayer.exe(Windows Media Player) を必ず閉じること。以上です。

test.ly の遊びは、<c' e' g'>1 の響きの c,g を半音高くして響きを聴いてみたのですが、これを、音楽的なややっこしい(難しい語で) 解説すると、音楽界のデフォルト常識である 4/4 拍子ハ長調ト音記号位置 c d g の和音響きを、イ長調で響かせ聴いてみた.....と言った説明になります。

LilyPond のデフォルト組版は、この文書の初期の 2 例で検証済みです。四分音符・4/4 拍子・ヘ音記号・ハ長調で組版出力されます。mid は指定楽器がなければ、"acoustic grand" (ピアノ) で作成されます。

理論は別として `\score{{...music code...}\layout{}\midi{}}` で*****.ly で save して、ly-run.cmd と実行すれば、譜面表示と mid が鳴り出します。
では、`a\major <cis' e' gis'>1` を trumpet で鳴らしてみます (test-2.ly)。

```
\score{{\key a\major\set Staff.midiInstrument="trumpet"<cis' e' gis'>1}\layout{}\midi{}}
```



実行は, (cmd.exe) >ly-run test-2[Enter] で.....自動車の警笛音響きに類似音でしょうか.

test.ly と **test-2.ly** のコーディングで半角空白の有無の差違が見てとられると思います. 通常は
`\key a \major \set Staff.midiInstrument ... \layout{} \midi{}` ですが,
`\key a \major \set Staff.midiInstrument ... \layout{} \midi{}` は許されます,
`\keya \major \set Staff.midiInstrument ... \layout{} \midi{}` は駄目です.
 "テキスト文字" or "\文字列" の後に明確である"\文字列"(命令語) 等々が続くときは, 半角空白は概ね省略できる.
 "\文字列" の後に"テキスト文字" が続く場合には, 明示的に命令語文字列は此処で終わりを知らせる為に, 半角空白が必要になります.
 また LilyPond のシーケンス文字列の内には半角空白を入れたシーケンス (配列綴り) が存在します, それも絶対に省略できません.
 コーディングに於ける改行は, 明示的に文字列 (単語) の終わりで改行です. テキスト文字列といえども文字列 (単語) の途中では改行をしないほうが良いです.
 (コード記述に於いて和語特有の全角空白は絶対に使用禁止です)

行内での % 以後一行以内 は LilyPond ではコメント開始で"%"以後の文字列は改行まで無視されます. 数行に渡るコメント記述は, 行頭を%{だけで改行します. 次行からコメント文章を開始し...コメント文最終で, 改行し次行頭で%}と宣言すれば, その間の記述は全て無視されます. これらは, 著者の覚書 (備忘録) 用です.

```
%{
July 31, 2009
編集; 段落を持つコメント文章の書き方
この様な, 記述になります. この空間を LilyPond の running は無視をします.
%}
```

LilyPond の解説書はそのバージョンによる, オンラインファイルである,
 LilyPond.pdf(印刷本)・LilyPond documentation(WEB ファイル) 等々を参照してください.
<http://lilypond.org/doc/v2.12/Documentation/>
 譜面知識さえ持っていれば, 英文の説明に頼らずとも LilyPond の作用と結果は自ずと読み取る事が可能になっています.

Windows OS の CLI(コマンドラインインターフェイス) は
 cmd 窓で >help > cmd-list.txt [Enter] で
 コマンドラインの命令語類とその解説が cmd-list.txt ファイル名で作成されますので, コマンドの検証はそちらで可能です (ユーザー入力 (二番目) の">"は OS のパイプ機能です).
 また, 命令の個別に欲しい時 (例えば copy コマンド) は,
 >help copy >> cmd-list.txt [enter] で, 上記の cmd-list.txt の最終行から追加上書きで取得できます. Windows OS の CLI 詳細理解追求は市販本でお願いします.

- LilyPond で遊ぶ楽しむ・ゲーム

Enjoy by LilyPond

As for coming next, what kind of sound?**Piano****Key e minor***your name*

date



この後に続くと言うか、来るべき音 (or メロディー) は.....? 人それぞれでしょう、貴方ならこの後をどう展開するか? (以下, Draft Source code)

```
%% -*- Coding: utf-8 -*-
\version "2.12.1"

\header {
  dedication = "Enjoy by LilyPond"
  title = "As for coming next, what kind of sound? "
  subtitle = "Piano "
  subsubtitle = " Key e minor "
  instrument = " "
  poet = ""
  composer = \markup{\italic "your name"}
  arranger = \markup{\tiny "date"}
}

play = { \autoBeamOff
%1-6
  \repeat volta 2 { r8 e'8_\pp fis'8[^\< a'] g'4 | b' c''8[ e''] d''4\! | }
  e'4 fis' g' | c'8[ e'16 g'] c''8[ b'] a'4 | fis'2.( | e'2.) |
\break
%%7- ここから描き始める

}

%% get music sheet -----
\score { \new Staff { \clef treble \key e\minor \time 3/4 \tempo 8 = 180
  \play
}
\layout {}
}

%% get music midi -----
\score { \new Staff { \clef treble \key e\minor \time 3/4 \tempo 8 = 180
  \unfoldRepeats \play
}
\midi {}
}
```

たとえば、ファイル名を play.ly と save して、既述の >ly-run play で譜面と mid を検証しながらコーディングを繰り返す、自分が納得できる作りが出来たらこのゲーム貴方の勝ちです!

私の描き上げた作品; <http://www.ipika.info/Poem/html/2007-mind2.html>

Concept Index

\

<code>\context Staff = "ID name" { ... }</code>	6
<code>\context Voice = "ID name" { ... }</code>	6
<code>\layout { ... }</code>	5
<code>\midi { ... }</code>	5
<code>\new Staff</code>	5
<code>\score</code>	5
<code>\version</code>	4

D

<code>dviout</code> のインストール	17
-----------------------------------	----

G

Ghostscript のインストール	17
---------------------------	----

L

LilyPond(Windows) インストール	15
--------------------------------	----

P

pTEX のインストール	17
--------------------	----

S

s 音符	7
------------	---

ま

マクロ記述方式 (方法)	4
--------------------	---

May it help you create lots of beautiful music!

The End

Windows 版 LilyPond はデスクトップ画面にもショートカットされています。このアイコンへ譜面原稿をスクロールすれば cmd.exe 経由で LilyPond は実行されます。

また、スタートメニューにも設定されていますが

しかし、これらは逆に Windows OS では使い難いですし、実用的ではありません。LilyPond の running は上記のようにコマンドライン入力実行インターフェースで使用が実践的です。

Uninstall 項目も有ります。

LilyPond の **Uninstall** は、コントロール・パネルの"プログラムの追加と削除"で LilyPond をクリック (削除表示ポップアップ) すれば OK です。後は空のフォルダ (残っているなら) を削除します。

Windows 版 LilyPond は cmd.exe を通して全て **LilyPond パッケージ内で働き Windows 側の物は利用しません**。譜面表示や midi の演奏はもちろん Windows 側へその手のアプリ (アクロバット・リーダー、メディアプレイヤー限定) へ渡します。

(LilyPond での和語環境は和語フォントの置き場所によります、自動で windows OS の fonts フォルダを検索してターゲット和語フォント名を探しますけど)

もし、gsview32.exe, gswin32.exe (T_EX 経由 dviout.exe,) 環境が存在すれば、これらでも譜面は表示可能です。mid 音源が存在するプレイヤーなら、その作成 mid もプレーします。

Windows OS 環境と言えども、Linux(unix) 系対応アプリケーションを構築するなら、環境設定で自己のホーム・ディレクトリを設定して置く事を推薦します。この場所にプログラム側が様々な run 参照するファイルを作成して置きます。

LilyPond に於ける記述コードは utf-8 コード記述を推薦です、"特に歌詞部分"は完全に utf-8 の言語コード指定ですので。

日本語 Windows OS 環境での system key 割り当て共通部分は以下を認識します。
 ASCII(American Standard Code for Information Interchange)
 ANSI(American National Standards Institute)
 Shift JIS(Japanese Industrial Standard)

楽譜や譜面, midi だけを作成取得なら,
LilyPond(ver2.10) での楽譜 (music sheet) や曲 (midi) のみを取得なら, 前ページの作業だけで目的は達します. また, その music sheet を Windows office ソフト等々のワープロ文書へも貼り付けは可能です.

下記のインストール作業は熟練を要しますし, 文書取得まで技術力が必要です. これらのオープンソフトプログラムは今日の電子情報と歴史的な印刷物情報の接点に位置しているソフトウェアだと言われています. また, 様々な学会 (世界の) への提出論文等々を作成するには, 必要になるアプリケーションです.

(下記に記載してある順番どおりにインストの事)

"角藤版 pTeX", "Ghostscript" と "dviout" (+ GSview) のインストール,
<http://www.nsknet.or.jp/~tony/TeX/texindex.html> TeX「超」入門
"TeX インストールガイド"をクリックして. 良く読んでからインストールしてください.
自己環境へは c:ドライブへインストールを推薦ですが, d:ドライブへインストールでも OK です.
pTeX パッケージは"最小"+"標準"で OK です, "GSview" のインストールは有れば極めて便利ですが, "GSview" はライセンスが必要 (ユーロ, ドル, カード決済 oK 約 3 千円以内?) です.
(注意; ユーロ, ドル決済の時, ユーロ圏各国・ドル圏各国での国々に於ける価格表示の単位区切り", ". "の日本社会との差異に十二分に注意を払ってください.)
(dviout のパラメータ値は前者 2 者を自動検索して, 自己設定します. TeX 専用です)
(GSview は Ghostscript の一般向け GUI です. Ghostscript は基本的には CLI でプロ仕様仕上げです. TeX とは関係有りません. Postscript 言語理解専用です.)

いずれにしましても, これらの初めてのインストールでも解説どおりに行えば, 殆どは間違いなくインストールとそれらの run は可能です.

続く,

