MusiXTEX©

Using TEX to write polyphonic or instrumental music

Version T.98 – January 18, 2001

Daniel TAUPIN
Laboratoire de Physique des Solides
(associé au CNRS)
bâtiment 510, Centre Universitaire, F-91405 ORSAY Cedex
E-mail : `taupin@lps.u-psud.fr`

Ross MITCHELL
CSIRO Division of Atmospheric Research,
Private Bag No.1, Mordialloc, Victoria 3195,
Australia

Andreas EGLER‡
(Ruhr–Uni–Bochum)
Ursulastr. 32
D-44793 Bochum

‡ *For personal reasons, Andreas Egler decided to retire from authorship of this work. Nevertherless, he has done an important work about that, and I decided to keep his name on this first page.* D. TAUPIN

If you are not familiar with TEX at all
I would recommend to find another software
package to do musical typesetting.
Setting up TEX and MusiXTEX
on your machine and mastering it
is an awesome job which gobbles up
a lot of your time and disk space.

But, once you master it...
HANS KUYKENS

[1]The word "free" meaning freely distributed and at no cost.

Contents

Chapter 1

What is MusiXTEX ?

Preface

Dear user,

This is still a provisional and imperfect version, which means that some features will be added or corrected. However our intent is to avoid removing existing commands, in order to spare your own adaptation problems or your typing time correcting scores you already have in your archive.

Sections marked with a star, or paragraphs marked "???" or ".!.", indicate either that text is partly missing, or that something might change at a next release.

MusiXTEX is a set of TEX macros to typeset polyphonic, orchestral or polyphonic music. Therefore, it is mainly supposed to be used to type wide scores – just because true musicians seldom like to have to frequently turn pages – and this is not really compatible with LATEX's standard page formats. Even with `A4.sty`, the `\textheight` and `\textwidth` are too small for musician needs.

However, a LATEX style has also been provided (and it is used for the typing of the present paper) but this `musixtex` style is fit for musicographic books rather than for normal scores to be actually played.

It should be emphasized that MusiXTEX is not intended to be a compiler which would translate some standard musical notations into TEX nor to decide by itself about aesthetic problems in music typing. MusiXTEX only typesets staves, notes, chords, beams, slurs and ornaments as requested by the composer. Since it makes very few typesetting decisions, MusiXTEX appears to be a versatile and rather powerful tool. However, due to the important amount of information to be provided to the typesetting process, coding MusiXTEX might appear to be as awfully complicated as the real keyboard or orchestral music. It should therefore be interfaced by some pre-compiler in the case of the composer/typesetter wanting aesthetic decisions to be automatically made by somebody (or something) else.

1.1 MusiXTEX principal features

1.1.1 Music typesetting is two-dimensional

Most people who have just learned a bit of music at college, probably think that music is a linear sequence of symbols, similar to literary texts to be TEX-ed. On the contrary, with the exception of strictly monodic instruments like most orchestral wind instruments and solo

note sequence one	note seq. four	note seq. seven	note seq. ten
note sequence two	note seq. five	note seq. eight	note seq. eleven
note sequence three	note seq. six	note seq. nine	note seq. twelve

Table 1.1: The order in which a musician reads music

voices, one should be aware that reading music is actually a matricial operation: the non-soloist musician successively reads *columns* of simultaneous notes which he plays if he is a pianist, clavichordist or organist, which he reads and watches if he conducts an orchestra, and which he is supposed to check and partially play when he is a soloist who wants to play in time with the accompanying instrument or choir.

In fact, our personal experience of playing piano and organ as well as sometimes helping as an alternate Kapellmeister leads us to think that music reading and composing is actually a slightly more complicated intellectual process: music reading, music composing and music thinking seem to be a three layer process. The musician usually reads or thinks several consecutive notes (typically a long *beat* or a group of logically connected notes), then he goes down to the next instrument or voice and finally assembles the whole to build a sequence of music lasting roughly a few seconds. He then proceeds to the next *beat* or *bar* of his score.

Thus, it appears that the humanly logical way of coding music consists in horizontally accumulating a set of *vertical combs* with *horizontal teeth* as described in Table 1.1. This is the reason why, in MusiXTEX the fundamental *macro* is of the form

`\notes ... & ... & ... \enotes`[1]

where the character `&` is used to separate the notes to be typeset on respective staffs of the various instruments, starting from the bottom.

In the case of an instrument whose score has to be written with several staffs, these staffs are separated by the character `|`. Thus, a score written for a keyboard instrument and a monodic instrument (for example piano and violin) will be coded as follows:

`\notes ... | ... & ...\enotes`

for each column of simultaneous *groups of notes*. It is worth emphasizing that we actually said *"groups of notes"*: this means that in each section of the previous macro, the music typesetter is welcome to insert, not only chord notes to be played at once, but small sequences of consecutive notes which build something he understands as a musical phrase. This is why note typing macros are of two kinds in MusiXTEX, namely the note macros which are not followed by spacing afterwards, and those which induce horizontal spacing afterwards.

1.1.2 The spacing of the notes

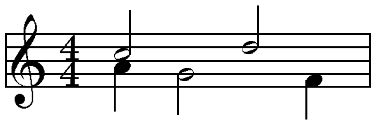
It seems that many books have dealt with this problem. Although it can lead to interesting algorithms, we think it is not an important point in practice.

In fact, each column of notes has not necessarily the same spacing and, in principle, this *spacing* should depend on the shortest duration of the simultaneous notes. But this cannot be established as a rule, for at least two reasons:

[1] Also available for `\enotes` is the abbreviation `\en`.

1. spacing does not depend only on the local notes, but also on the context, at least in the same bar.

2. in the case of polyphonic music, exceptions can easily be found. Here is an example:



where it can be clearly seen that the half notes at beats 2 and 3 must be spaced as if they were quarter notes since they overlap, which is obvious only because of the presence of the indication of the *meter* 4/4.

So the typesetter has to take care of good readable spacings on his own. Therefore, we preferred to provide the composer/typesetter with a set of macros which can be found in 2.3.2.

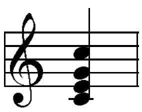
1.1.3 Music tokens, rather than a ready-made generator

The tokens provided by MusiXTEX are:

- the note symbols *without stem*;
- the note symbols *with stems, and hooks for eighth notes and shorter*;
- the indication of beam beginnings and beam ends;
- the indication of beginnings and ends of ties and slurs;
- the indication of accidentals;
- the ornaments: arpeggios, trills, mordents, pincés, turns, staccatos and pizzicatos, fermatas, etc.;
- the bars, the meter and signature changes, etc.

Thus, `\wh a` produces an *A (of nominal frequency 222.5 Hz, unless transposed)* of duration being a *whole note*. In the same way, `\wh h` produces an *A (445 Hz)* of duration represented by a *whole note*, `\qu c` produces a *C (250 Hz approx.)* whose value is a *quarter note with stem up*, `\cl J` produces a *C (125 Hz approx.)* of duration equal to an *eighth note with stem down*, etc.

To generate quarter, eighth, sixteenth, etc. chords, the macro `\zq` can be used: it produces a quarter note head, the vertical position of which is memorized and recalled when another stemmed note (possibly with a hook) is coded; then the stem is adjusted to link all simultaneous notes. Thus, the perfect C-major chord, i.e.



is coded `\zq c\zq e\zq g\qu j` or, in a more concise way, `\zq{ceg}\qu j` (stem up): in fact, single notes are treated... like one-note chords.

1.1.4 Beams

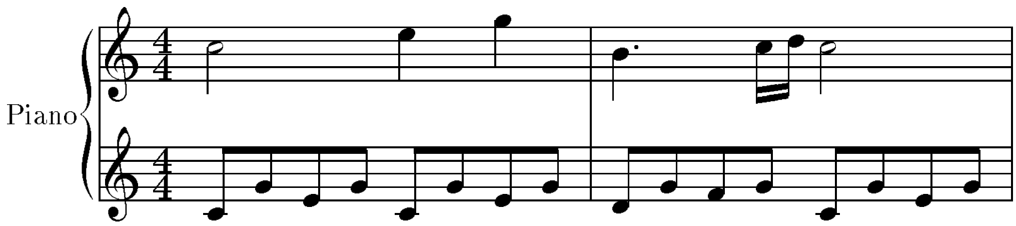
Beams are generated using macros which define their beginning (at the current horizontal position), together with their altitude, their direction (upper or lower), their multiplicity, their slope and their reference number. This latter feature – the reference number – appears to be necessary, since one may want to write beams whose horizontal extents overlap: therefore, it is necessary to specify which beam the notes belong to and which beam is terminated at a given position.

1.1.5 Setting anything on the score

A general macro (`\zcharnote`) provides a means of putting any sequence of symbols (in fact, some `\hbox{...}`) at any pitch of any staff of any instrument. Thus, any symbol defined in a font (letters, math symbols, etc.) can be used to typeset music.

1.2 A simple example

Before giving more details, we give below an example of the two first bars of the sonata in C-major KV545 by MOZART:



The *coding* is set as follows:

```
\begin{music}
\parindent10mm
\instrumentnumber{1}       % a single instrument
\setname1{Piano}           % whose name is Piano
\setstaffs1{2}             % with two staffs
\generalmeter{\meterfrac44}% 4/4 meter chosen
\startextract              % starting real score
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\ql l\sk\ql n\en
\bar
\Notes\ibu0f0\qb0{dgf}|\qlp i\en
\notes\tbu0\qb0g|\ibbl1j3\qb1j\tbl1\qb1k\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\endextract                 % terminate excerpt
\end{music}
```

- `\ibu0f0` begins an upper beam, aligned on the f, reference number 0, slope 0
- `\tbu0` terminates this beam before writing the second g by means of `\qb0g`
- `\qb..` indicates a note belonging to a beam.

- `\sk` sets a space between the two quarters in the right hand, so that the second is aligned with the third eighth of the left hand.
- `\qlp` is a quarter with a point.
- `\ibbl1j3` begins a double beam, aligned on the C (`j` at this pitch) of slope 15%.

1.3 The three pass system

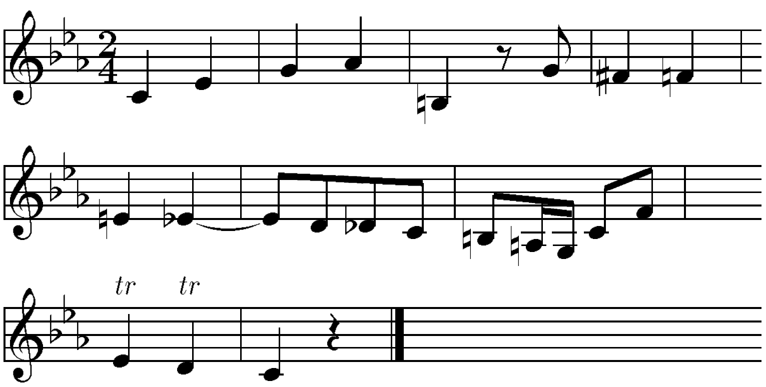
TEX's line-breaking procedure implicitly assumes that a normal line of text will contain many words, so that inter-word glue need not stretch or shrink too much to justify the line. This strategy does not work very well for music. If each bar of music is treated as a word, in the sense that inter-bar glue is placed at the end of each bar, then the usual result is the appearance of unsightly gaps before each bar rule. This follows naturally from the fact that the number of bars per line is normally many fewer than the number of words in a line of text.

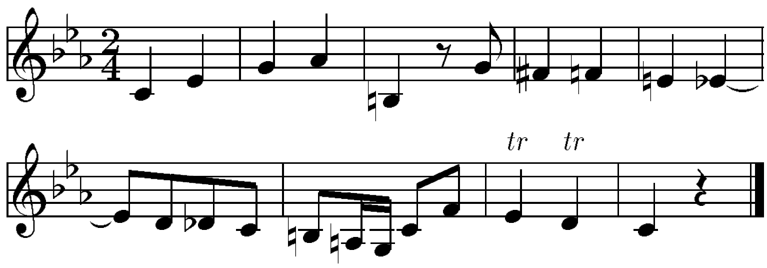
To address the above aim, a three pass system was developed. On the first pass, information about each bar is written to an external file, by default `jobname.mx1`. This file begins with a header listing parameters such as line width, paragraph indentation, clef, sign and meter widths, and the first-pass elemental spacing `\elemskip`. Width information within the bars is classified either as "soft" or "hard": "soft" widths are those amenable to subsequent rescaling (e.g. a note box), while "hard" widths represent unscalable material such as clef symbols, key signatures and bar rules.

After the first pass, an external routine is run to determine optimal values of the elemental spacing (`\elemskip`) for each line, so as to properly fill each line, and to lead to the piece filling an integral number of lines. This routine was written in FORTRAN and now converted to C rather than TEX, the main reason being the lack of an array handling capability in TEX, and the concern that if such a routine were added to MusiXTEX, storage problems might be exacerbated.

The resulting routine reads in the file `jobname.mx1`, and writes its output to `jobname.mx2`. The latter file contains a single entry for each line of music in the reformatted output. The key piece of information is the revised value of `\elemskip` for each line.

Next, the file is re-TEX-ed. On this third pass, the `jobname.mx2` file is read in, and the information used as described above.

To give an easy example: after pass 1 you get the following output:

After running `musixflx` and TEX-ing the second time you'll get:



which was coded as:

```
\hsize=100mm
\generalmeter{\meterfrac24}%
\parindent 0pt
\generalsignature{-3}
\startpiece\bigaccid
\NOtes\qu{ce}\en\bar
\NOtes\qu{gh}\en\bar
\NOtes\qu{=b}\en
\Notes\ds\cu g\en\bar
\NOtes\qu{^f=f}\en\bar
\NOtes\qu{=e}\itied0e\qu{_e}\en\bar
\Notes\ttie0\Qqbu ed{_d}c\en\bar
\Notes\ibu0b{-2}\qb0{=b}\enotes
\notes\nbbu0\qb0{=a}\tqh0N\enotes
\Notes\Dqbu cf\en\bar
\NOtes\uptext{\it tr}\qu e\uptext{\it tr}\qu d\en\bar
\NOtes\qu c\qp\en\Endpiece
```

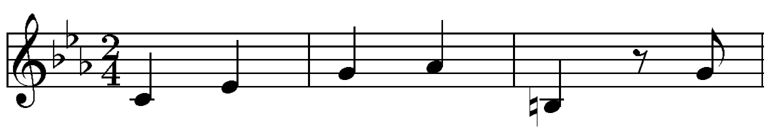
The biggest advantage of using a 3-pass system is the very easy and fast alteration to the layout which is possible, especially of a long masterpiece, by changing only one parameter, namely `\mulooseness`. This value acts analogous to TEX's `\looseness` command. For non-TEX-perts: if you end a paragraph and state `\looseness=-1` somewhere inside, then TEX will try to make this paragraph one line shorter than it normally would.

MusiXTEX does the same, but instead for paragraphs, for *sections*. What is a *section*? As long as you don't force a line break, this means the whole piece. When you force a line break with either `\stoppiece`, `\endpiece`, `\zstoppiece`, `\Stoppiece`, `\Endpiece`, `\alaligne` `\zalaligne`, `\alapage` or `\zalapage` you start a new *section*. Somewhere[2] before the end of the *section*, you can change the value of `\mulooseness` to something different from the default of 0.

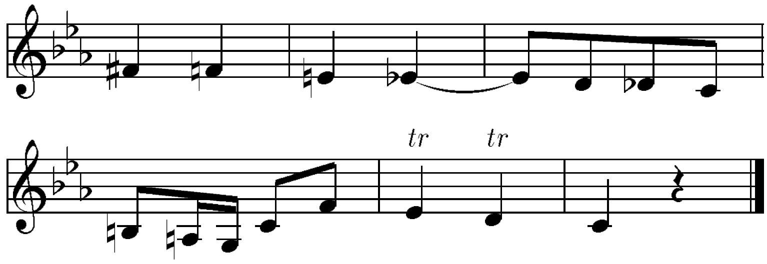
To give an easy example, changing the last line in the previous example to:

```
\NOtes\qu c\qp\en\mulooseness=1\Endpiece
```

yields:



[2]Advisably, at the beginning or at the end of the section, for the sake of clarity.



or:

```
\NOtes\qu c\qp\en\mulooseness=-1\Endpiece
```

yields (a little bit tight, but only for the sake of demonstration of `\mulooseness`):

Here is the fastest way to get a satisfactory output:

1. build up the piece bar[3] by bar. Pay much attention to correct spacings, and go on only when you are really satisfied with them.

2. TEX $\Longrightarrow$ `musixflx` $\Longrightarrow$ TEX.

3. Look at the output and decide if you want to have more or fewer systems lines, e.g. to fill the page or to get an even number of pages.

4. Adjust `\mulooseness`. However, **beware :** `\mulooseness` increases the target number of lines by its value. This is of no serious trouble for large pieces, but if you say `\mulooseness=3` in a piece or in a section which only has 3 bars (three in the musical sense, i.e. two `\bar` in the text), then `musixflx` will not be able to split 3 bars into 4 lines... and it will hopefully result in a diagnostic by `musixflx`, but possibly also in an arithmetic fault...

5. Delete the `jobname.mx2` and again
   TEX $\Longrightarrow$ `musixflx` $\Longrightarrow$ TEX *(For hackers: watch the difference in* `jobname.mx1`*!)*

REMARK: Instead of looking at the output of the previous run and modifying the number of systems using `\mulooseness` *one can directly specify the number of lines in a section by assigning a positive number to* `\linegoal`*. Use* `\linegoal` *in the same way as* `\mulooseness` *somewhere in a section, preferable at start or end.* `\mulooseness` *must be zero for* `\linegoal` *to work. Both are automatically set to zero after processing the end of a section e.g. by* `\stoppiece`*.* `\linegoal` *requires the latest version of musixflx (0.83).*

If you have really followed this step by step, then your whole layout will be ready and you will not have to generate `jobname.mx2` again. This only holds as long as you don't change any spacings.

If you get the warning: `You can't use \raise ...`, than read the upper paragraph again and follow the instruction of point 5.

There are some more advantages. Without using *glue*, every horizontal position in a line can be computed. This enables using, e.g. octave lines, without specifying a dimension, but by saying `\i...` and `\t...`.

[3] Or two or three, whatever you prefer.

1.3.1 External executable `musixflx`

One drawback affecting portability between computers is the executable `musixflx`. We are still searching for people who are able to compile the C-source and maintain it.

On most computers, the executable is invoked by typing the name of the program and the name of the file to be acted upon. *i.e.*

`musixflx jobname.mx1`

Optionally, you can add a letter to indicate one of the debug modes, which are:

- `d` for debug information to screen
- `f` for debug information to file `jobname.mxl`
- `s` to get the computed lines immediately on screen

To allow for ease of use with a batch file, `musixflx` can either be fed with `jobname.mx1`, `jobname.tex` or only `jobname`, all of which opens `jobname.mx1`.

For large scores (more than 4 pages approximately), having only one section and an overall value of `\mulooseness` becomes impractical since one wants not only to have nicely spaced systems, but one also usually wants to have completely filled pages to avoid empty top and bottom margins on the last page. It is then wise to force the total number of pages and maybe the line breaks in each page, which can be done by either using explicit `\alapage` and `\alaligne`, or more automatically by means of the `\autolines` command borrowed from MusicTEX and implemented in the additional `musixcpt.tex` file.

1.3.2 Restrictions and warnings

If you have worked with MusicTEX before, you must be aware that two main things have changed, namely the spacing (and spacing commands) and the line breaking, see section 2.8, p. 33 and section 2.14.1 p. 48.

A serious practical problem concerns the effect of unrecorded spaces, such as those created by source lines not ended by `%` or `\relax` or another control sequence ending in a letter. These will result in the dreadful **Overfull hbox** messages appearing during pass 3.

Considerable discipline is needed to avoid this problem!

When you start using this software package, you will spend a lot of time searching for the cause of over/underfull-hbox warnings, which will appear during pass 3. You may ignore them during pass 1, but *you must* find their cause when they appear during pass 3.

In MusiXTEX there is no glue (only ± 1pt behind `\writesignatures`), so it works only if every space is recorded in the correct way. As a result, never use `\hskip` or `\kern` with the exception of *zero*-boxes, like `\rlap`, `\llap`, `\zcharnote`, `\uptext`, etc., or (apparently) fixed length boxes like `\hsong`. But that is not all—all spacing commands[4] must be fed with scalable values (*e.g.* `\noteskip`, `\elemskip`, `\afterruleskip` and `\beforeruleskip`, optionally preceded by an integer or decimal number).

`musixflx` can't work if the values of `\noteskip`, `\afterruleskip`, `\beforeruleskip`, are directly assigned "hard"[5] values in between `\startpiece` and `\stoppiece`[6].

These restrictions make it hardly compatible with most sources designed for MusicTEX[7]. Most of them have to be adjusted to get things going.

[4] Except `\hardspace` which is an exception and has the opposite purpose.

[5] Hard values are all values which are not scalable

[6] `\endpiece` has the same meaning

[7] Most examples provided by D. TAUPIN in MusicTEX's distribution can be run with MusiXTEX without serious problems... However, they will eventually have to be corrected for the sake of compatibility.

Some values are much more important than in MusicTEX, namely `\elemskip`, `\afterruleskip` and `\beforeruleskip`. The ratio between them is especially very important if you use `\mulooseness` often to adjust the number of lines: all of them are multiplied by the same factor in each section.

1.3.3 MusiXTEX's laws

1. Never have a line that doesn't finish either with a control sequence or with a `%` inside of `\startpiece...\stoppiece`

   !!! MusiXTEX forgives NOTHING !!!

   (a space in the default font (using plain-TEX) causes 3.33pt of unrecorded space, but there is only $\pm$ 1pt of glue in MusiXTEX)

2. `\off` must be used with relative values, like `\noteskip`, `\elemskip`, `\after[before]ruleskip`

3. `\qsk` and `\hqsk` are now scalable, e.g. `\qsk` doesn't mean exactly one note head width (it depends on `\elemskip` and `flex_factor`)

4. Text of songs and any non-MusiXTEX text must be put in zero boxes, like `\zcharnote`, `zchar`, `rlap`, `lrlap`, `llap`, `uptext`, `zsong....` However, specific features are now provided, namely `\hardlyrics` and `\hsong` (see 2.20.7, p. 70).

5. Don't touch either `\noteskip` with hard values, or `\after[before]ruleskip` at all inside of `\startpiece (\debutmorceau) ... \stop[end]piece (\susp[fin]morceau)`

1.4 Some highlights

1.4.1 Signatures

Signatures can be stated either for all instruments, for example by `\generalsignature{-2}` which sets two flats on each staff, or separately for each instrument.

Thus, the `\generalsignature` can be partly overridden by `\setsign2{1}` which puts one sharp on the staffs of *instrument number* 2. Of course, the current signature may change at any time as well as the meters and clefs.

1.4.2 Transposition

An important question is: *"can MusiXTEX transpose a score ?"*. The answer is now 99.5 % *yes*. If fact, there is an internal register named `\transpose` the default value of which is zero, but it may be set to any reasonable positive or negative value. In that case, it offsets all symbols pitched with letter symbols by that number of pitch steps. However, it will neither change the signature nor the local accidentals, and if – for example – you transpose a piece written in C by 1 pitch, MusiXTEX will not know whether you want it in $D\flat$, in D or in $D\sharp$. This might become tricky if accidentals occur within the piece, which might have to be converted into flats, naturals, sharps or double sharps, depending on the new chosen signature. To avoid this trouble, *relative* accidentals have been implemented, the actual output of which depends on the pitch of this accidental and on the current signature.

1.4.3 Selecting special instrument scores

Another question is: *"can I write an orchestral score and extract the separate scores for individual instruments ?"* The answer is 95 % *yes*: in fact, you can define your own macros `\mynotes...\enotes`, `\myNotes...\enotes` with as many arguments as there are in the orchestral score (hopefully this is less than, or equal to 9, but TEXperts know how to work around) and change its definition depending on the selected instrument (or insert a test dependent on the value of some selection register). But the limitation is that the numbering of instruments may change, so that `\setsign3` may have to become `\setsign1` if instrument 3 is alone. But, in turn, this is not a serious problem for average TEX wizard apprentices.

1.4.4 Variable staff and note sizes

Although the standard staff size is 20pt, MusiXTEX allows scores of 16pt and 24pt staff sizes. In addition, any instrument may have a special staff size (usually smaller than the overall staff size) and special commands `\smallnotesize` or `\tinynotesize` enable notes (and also beams or accidentals) to be of a smaller size, in order to quote optional notes or *cadenzas*.

1.5 How to get it

Primary site is `hprib.lps.u-psud.fr` (anonymous `ftp`); say:

`cd pub/musixtex`

`mget musix*.tex` (to get the macros)

`cd /pub/musixtex/mf`

`mget *.mf` (to get the source of the fonts)

`cd /pub/musixtex/tfm`

`binary`

`mget *.tfm` (to get the font metrics)

or:

`cd /pub/music\_zips`

`binary`

`mget musix*.zip` (to get the zipped packages specific to MusiXTEX)

Official mirrors are the CTAN servers. To get it you have tou say:

`cd ctan` (to go to the CTAN server)

`cd macros/musixtex/taupin`

`binary`

`mget *.zip` (to get the zipped packages)

The whole *distribution* can fit into a single 1.2Mbyte or 1.44Mbyte diskette. All sources (including fonts) are provided, either separately or "zipped".

REMARK: normally, the `musixtex.zip` *file expands into binary files valid for any system, and text files (the TEX and* METAFONT *sources) in the* UNIX *representation, namely with a "LF" character between each record. PC users might complain that their own coding is "CR+LF" ("CR" usually appears as a "*`^M`*"), but we checked that:*

- *the MS-DOS editor handles both denotations,*

- *the "emacs" and "microemacs" also do,*
- *the TEX and* METAFONT *executable routines (et least the emTeX ones) also accept both representations.*

However, for the sake of compatibility, we now provide two routines for PC's, namely `dtou.exe` *and* `utod.exe`*, which convert text/source files from/to the MS-DOS denotation to/from the* UNIX *denotation.*

1.6 Enhancements

1.6.1 Recent easy enhancements

Many enhancements have been asked for, to the "father" of MusiXTEX, namely MusicTEX, and this is a proof that it is considered as useful by many people. Some of these enhancements which seemed hard were in fact rather easy to implement, for example small notes to represent grace notes and cadenzas, or gregorian chant features.

Some of them were more difficult, such as handling of lyrics now provided since version T.40.

1.6.2 Enhancement limitations

Many requested improvements have not been *yet* implemented for several reasons:

- The authors' natural laziness (!)
- More seriously: many of them would require using some more registers; unfortunately, TEX registers are not numerous (256 of each kind and the limit of `\dimen` registers is nearly reached (especially using LATEX), but we do our best).
- We do not think it is wise to introduce in MusiXTEX itself a great number of macros which would be poorly used by most users: the reason is that TEX memory and — moreover — TEX registers are severely limited and that unused macros may occupy many of these, leading to things crashing because of `TeX capacity exceeded`.

  Therefore, many additional library files are now provided, which may be invoked or not, depending on the user's specific needs. Most of them are used to compile this LATEX 2_ε manual... except the `musixadd.tex` and `musixmad.tex`.

1.7 Acknowledgements

The idea of implementing the present package is due to the previous work (MuTEX) of Andrea STEINBACH and Angelika SCHOFER[8]. This work provided the basis of the Metafont codes which are still used here... with 100% corrections and updates.

English language is progressively checked thanks to Cecil CHURMS from National Accelerator Center (South Africa).

[8]Steinbach A. & Schofer A., *Theses* (1987, 1988), Rheinische Friedrich-Wilhelms Universität, Bonn, Germany.

Chapter 2

Practical use

2.1 Heading statements

Before any reference to MusiXTEX macros:

`\input musixtex`[1]

which may be followed by `\input musixadd` in the case you have more than six instruments or more than 6 simultaneous beams or ties or slurs.

After that, you may write a complete book of TEX provided that you do not use `&` as a tabulation character (its `\catcode` has been changed) inside the music score and that you do not overwrite MusiXTEX's definitions. This means that no special macros have been designed to help you write titles, author names, comments, literature excerpts, etc., unless you use LaTeX with the `musixtex` style.

2.2 Before you begin to write notes

2.2.1 Warnings for the non TEXpert

When TEX reads a parameter, it inputs one *token*. This means *one* figure or *one* character or *one* command[2]. If you want to give more than one char or e.g. a negative number('-' is one token and a digit is one) or a command which reads another parameter, then you have to use braces to transfer the parameter correctly. For example, the command `\meterfrac` reads two parameters; if you want to declare a $\frac{8}{12}$ timing and you code: `\meterfrac812`, then you will get a $\frac{8}{1}$ timing and a misplaced "2". The correct coding is `\meterfrac{8}{12}` or `\meterfrac8{12}`.

The next problem is due to spaces. Spaces are ignored at the beginning of a line. Spaces are ignored immediately behind a TEX-command (they indicate the end of that command unless TEX finds characters which are not *letters*). All other spaces insert *unexpected* space and they will cause a lot of trouble during third pass. Same remark holds for end of lines. Thus `\bar` or `\enotes` as last statement in current line cause no trouble, because an end of line indicates to TEX the end of a command.

[1]It is highly recommended to build up a format. If you don't know how to do, you can refer to the same instruction you perhaps heard or read before: "Ask you local TEX-wizard.". Otherwise it might be a good idea to look at chapter 3 ("Installation") where some hints are given to build a MusiXTEX format (refformatbuild).

[2]Syntactically, a *command* is made of a *backslash* ("\") immediately followed by sequence of *letters*.

If you state `\qu g` and start a new line, then you get into trouble, but if you say `\qu g%`< *newline* >< *spaces* >, you have no more problem and this is better readable.

If this sounds complicated, remember that TEX was designed to typeset text and not music...

2.2.2 What you have to specify

You should first specify whether you want to typeset music in "normal" size (20pt per staff) or "small" size (16pt) or "large" size (24pt). This is only optional, the default setting is `\normalmusicsize`. If you want the smaller size, then you have to say[3] `\smallmusicsize`, `\largemusicsize`, and `\Largemusicsize` to have the larger sizes of 24pt and 28.8pt.

Then, the first compulsory declaration is:

`\instrumentnumber{`n`}`

where n is the number of instruments, used by MusiXTEX to performs loops building staffs, setting signatures, meters, etc. Therefore, it must be defined before any other statements. If not then default value is one. An instrument may consist of several staffs, e.g. the piano. The difference between one instrument of several staffs and several instruments is as follows:

- distinct instruments may have distinct *signatures*, distinct staffs of a unique instrument share the same signature.
- *stems* may be hung to *beams* belonging to differents staffs of the same instrument.
- *chords* may extend across several staffs of the same instrument.
- staffs of a unique *instrument* are tied together with a big brace at the beginning of each line.

If the number of staffs is not equal to one, this number must be specified by:

`\setstaffs` n`{`p`}`

where p is the number of staffs, and where n is the number of the instrument considered (e.g. `setstaffs3` for the 3rd instrument, starting from the bottom).

Unless all your instruments only use the *violin* clef, you have to specify all the clefs used for all the instruments. This is done by coding:

`\setclef{`n`}{`$s_1 s_2 s_3 s_4$`}%`

where n is the number of the instrument, s_1 is one figure specifying the clef of the lower staff, s_2 the clef of the second staff, etc.

The value $s_1 = 0$ means the *violin* clef (G clef or *clef de sol* in French) for the lower staff, $s_2 = 0$ means the violin clef for the second staff, etc. If s_2, s_3 or s_4 are omitted in the second argument of the `\setclef`, these staffs have de default clef, namely 0, i.e. the violin clef.

The values $s_1 = 1$ through $s_1 = 4$ mean the *alto* clef (*clef d'ut* in French) set on first (lower) through fourth (next to upper) line of the staff.

The value $s_1 = 5$ means the *bass* clef at third (middle) line, $s_1 = 6$ means the usual *bass* clef (F clef or *clef de fa* in French) at the usual fourth line, and $s_1 = 7$ means the *bass* (*subbass*) clef (F clef or *clef de fa* in French) at the fifth line.

The value $s_1 = 8$ is not used.

[3]This sets up not only the size of staff, but also the elementary spacing and the space behind the bar rule.

The value $s_1 = 9$ means the *french violin clef*, an ancient feature which sets the G clef (*clef de sol* in French) at the fist line of the staff.

If remembering the digits associated with usual clefs seems too difficult, one can also say

```
\setclef{n}{\treble}
```

to get the violin clef on all staffs of the specified instrument. In the same way the most used *alto clef* (alto clef on 3rd line) can also be called with

```
\setclef{n}{\alto}
```

which sets the lower staff of thet instrument to the alto clef, but sets other staffs above to the violin clef. The usual *bass clef* on fourth line can also be called with

```
\setclef{n}{\bass}
```

IMPORTANT: the commands `\alto` *and* `\bass` *only set these clefs at the first staff. Other staffs still have the default value of violin clef.* As an example, a standard piano score should include:

`\setclef1{\bass}%` or `\setclef1\bass`

taking account of the fact that the bass clef is set on the lower staff, but the upper one has the default value of violin clef. If the signature is not void, one should code:

```
\generalsignature{s}
```

where $s > 0$ indicates the number of *sharps* in the signature and $s < 0$ the number of *flats*[4]

If a *meter* indication is to be posted, it should be specified using the macro

```
\generalmeter{m}%
```

where m is the description of the meter indication which should appear on each staff. If it is a *fraction* (e.g. 3/4) on should code

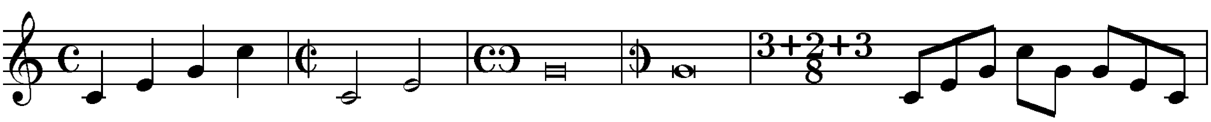
```
\generalmeter{\meterfrac{3}{4}}%
```

or, in a simpler way (if the numbers are less than 10):

```
\generalmeter{\meterfrac34}%
```

You can suppress the beginning vertical rule with saying `\nostartrule` and restore the default with `\startrule` after that.

Special denotations can be used, such as `\meterC`, `\allabreve`, `\reverseC`, `\reverseallabreve` and `\meterplus`.



which was coded as:

```
\generalmeter\meterC
\nostartrule
\parindent0pt\startpiece
\NOtes\qa{cegj}\enotes
\generalmeter\allabreve\changecontext
\NOTes\ha{ce}\enotes
\generalmeter\reverseC\changecontext
\NOTEs\zbreve g\enotes
\generalmeter\reverseallabreve\changecontext
```

[4]We have seen once a score in G-minor where the signature consisted of two flats (B and E) plus one sharp (F). This is not directly supported by MusiXTEX.

```
\NOTEs\zwq g\enotes
\generalmeter{\meterfrac{3\meterplus2\meterplus3}8}\changecontext
\Notes\Tqbu ceg\Dqbl jg\Tqbu gec\enotes
\endpiece
```

However, not all music scores have the same meter in each staff. Especially, some staffs may have *ternary* meters while others have *binary*. This can be specified by using the `\generalmeter` macro to set the meter for most of the scores and overriding it by:

`\setmeter2{{\meterfrac{12}8}\allabreve}%`

which sets the meter to 12/8 for the first (lower) staff, and *alla breve* for the second staff of the instrument number 2.

Sometimes it might be useful to insert additional space before the meter is written. This is possible by assigning a value to `\meterskip` in the preamble. It is reset to zero after first meter indication.

2.2.3 Instrument names

If you want the *name of the instruments* (or the *name of the voices*) to be displayed in front of their respective staffs at the beginning, you may code:

`\setname` n{*name of the instrument*}%

where n is the number of the instrument considered. In this case, you should also adjust the `\parindent` dimension so that the long name of an instrument does not spill too far into the left margin.

2.2.4 Polyphonic songs

Except staffs of a unique instrument tied together with a big brace, staffs normally begin on the left with a thin vertical rule. However, it is usual to tie all human voices together with a left heavy and right thin vertical rule.

2.2.4.1 Case of only one choir

In the usual case where you have only one choir, this can be specified by:

`\songtop{`n`}`
`\songbottom{`m`}`

where m and n are the instrument numbers of the first and last choral voices. Example is shown in 2.26.4.

No heavy square brace is produced when the bottom instrument has a number greater than the top one. This feature can be used to remove choir heavy braces between distinct parts of the same score.

2.2.4.2 Case of several choirs or several orchestras

Your may divide your instruments into up to three groups linked together with a left square brace. Each of these groups is specified with:

`\grouptop{`g`}{`n`}`
`\groupbottom{`g`}{`m`}`

where m and n are the instrument numbers of the first and last choral voices of group number g. MusiXTEX allows up to three groups, numbered from 1 to 3, plus the "zero" group consisting of the voices or instruments which belong to no declared group. `\songtop` is equivalent to `\grouptop 1` and `\songbottom` is equivalent to `\groupbottom 1`.

With `musixadd.tex`, this number of groups is raised to four.

This grouping feature can also be used to collect not only voices but also instruments with one or several staffs. In that case the normal brace is shifted left of the square havy brace.

2.2.5 Removing square choir braces

Previously defined square choir braces can be removed by stating a `\songtop` less than the `\songbottom`. The same applies to `\grouptop` and `\groupbottom` of the same group number.

2.2.6 Specifying all brackets together

Alternatively, you can use the following command for specifying all required brackets together:

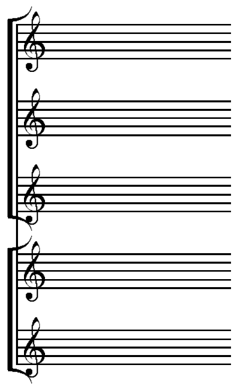
\akkoladen{{*lower_1*}{*upper_1*}{*lower_2*}{*upper_2*}{*lower_3*}{*upper_3*}}

where *lower_n* and *upper_n* are instrument numbers that denote the span of bracket number n. For setting fewer than three brackets, just omit all unneeded {*lower_n*}{*upper_n*} pairs; `\akkoladen{}` cancels all brackets.

Example:

```
\instrumentnumber{5} \akkoladen{{1}{2}{3}{5}}
```

yields:



which was coded as:

```
\smallmusicsize\setsize1\smallvalue\setsize2\smallvalue
\setsize3\smallvalue\setsize4\smallvalue\setsize5\smallvalue
\instrumentnumber{5} \akkoladen{{1}{2}{3}{5}}
\startextract \hardspace{2cm}\zendextract
```

2.3 Starting your masterpiece

2.3.1 Typing the first system

Just code `\startmuflex`

which opens the *jobname*`.mx1` during TEXing the first time your masterpiece. All informations are now written to it. During pass 3 it will be open for read in and all computed values are used to set the lines[5].

REMARK: do not worry too much if you forget `\startmuflex`*; in most cases — since version T.39 — the* `\startpiece` *command will do it, if necessary. Conversely, only the first* `\startmuflex` *is effective, the others are inefficient... for safety.*

Then code

`\startpiece`

which will initiate (with indentation `\parindent`) the first set of staffs for all instruments you have previously defined. But that is not sufficient to begin writing notes and silences. In fact, you must also choose the spacing of the notes.

2.3.2 Easy selecting note spacing

Therefore, we preferred to provide the composer/typesetter with a set of macros (incidentally, this can be adjusted):

usage	spacing	suggested use for
`\znotes ... & ... & ... \enotes`	(non spacing)	specials
`\notes ... & ... & ... \enotes`	`2\elemskip`	𝅘𝅥𝅯 16th
`\notesp ... & ... & ... \enotes`	`2.5\elemskip`	𝅘𝅥𝅯. 16th pointed, 8th triplet
`\Notes ... & ... & ... \enotes`	`3\elemskip`	𝅘𝅥𝅮 8th
`\Notesp ... & ... & ... \enotes`	`3.5\elemskip`	𝅘𝅥𝅮. 8th pointed, 4th triplet
`\NOtes ... & ... & ... \enotes`	`4\elemskip`	𝅘𝅥 4th
`\NOtesp ... & ... & ... \enotes`	`4.5\elemskip`	𝅘𝅥. 4th pointed, 2th triplet
`\NOTes ... & ... & ... \enotes`	`5\elemskip`	𝅗𝅥 2th
`\NOTesp ... & ... & ... \enotes`	`5.5\elemskip`	𝅗𝅥. 2th pointed
`\NOTEs ... & ... & ... \enotes`	`6\elemskip`	𝅝 1th

If you don't like them, define your own using e.g.:

`\def\NOtes{\vnotes5.34\elemskip}`

which means, when you use this `\NOtes` definition, that all *spacing* notes and commands, results a spacing of $5.34 \times$ `\elemskip`. The size of the spatial unit (`\elemskip`) can be freely adjusted (it shows only effect outside `\startpiece...\endpiece`).

In practice, the choice of the macro `\notes`, `\Notes`, `\NOtes`, etc., to initiate of column of notes sets an internal dimension register, named `\noteskip` to the given multiple of `\elemskip`. Thus, each *spacing note* (`\qu`, `\qb`, `\hl`, etc.) will be followed by a spacing of `\noteskip`. Then, the advantage of the definition of `\elemskip` is that, whenever it is changed, all subsequent `\noteskip`s will be updated proportionally so that a simple change of `\elemskip` can expand or shrink all consecutive note spacings as a whole.

All said above counts only for first pass.

[5]The files must be closed before leaving TEX, preferably before `\bye` or `\end`, with `\endmuflex`. Normally TEX closes all open files on his own when terminating the program, but it is still wiser to do this explicitly.

If the arthmetic progression of note spacings does not meet your wishes, your may force a geometric progression, where `\Notes` is $\sqrt{2}$ times wider as `\notes`, `\NOtes` is $\sqrt{2}$ times wider as `\Notes`, etc. In that case, `\Notesp` is approximately $\sqrt{\sqrt{2}} = 1.189$ times wider as `\Notes` and so on. The geometric progression if forced by the command `\geometricskipscale`, and the original arithmetic progression is restored by `\arithmeticskipscale`.

Inside the pair `\notes...\enotes` you may freely change the value of `\noteskip` (not `\elemskip`) provided you keep it scalable in the normal situation[6]. `\noteskip` can be changed with a command like

```
\noteskip=2.4\noteskip
```

which obviously keeps it scalable, or by means of

```
\multnoteskip{2.4}
```

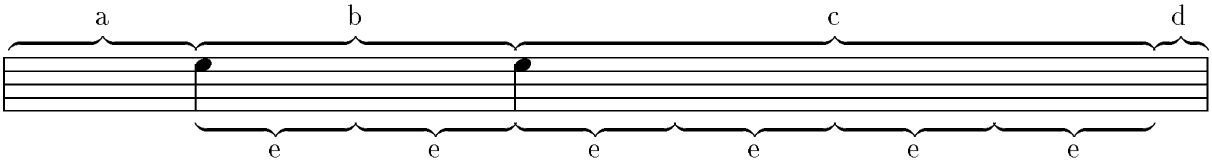
which performs the same multiplication in a smarter way. You can also force all `\noteskip`s to be re-scaled by a factor of 3.1415 using

```
\scale{2.4}
```

or `\def\scalenoteskip{2.4}` which is more dangerous and not advises although compatible with MusicTEX.

To fill the music line (system) without glue, the values of `\elemskip`, `\afterruleskip` and `\beforeruleskip` are adjusted from the external program `musixflx`. So what spacing you really get, depends on the internal line-breaking routine.

Spacings for advanced MusiXTEX'ers: a 'normal' bar (slightly !) magnified:



- a → `\afterruleskip`
- b → `\notes = \vnotes 2\elemskip`
- c → `\NOtes = \vnotes 4\elemskip`
- d → `\beforeruleskip`
- e → `\elemskip`

As default the values are set up as follows:

using	`\elemskip`	`\afterruleskip`	`\beforeruleskip`
`\normalmusicsize`	6pt	8pt	0pt
`\smallmusicsize`	4.8pt	6pt	0pt

If you want to change upper values, you have to do that behind `\normalmusicsize` or `\smallmusicsize` and before `\startpiece`.

2.4 Note pitch specification

Note pitches are usually specified by letters ranging from `a` to `z` for those which are usually written under the G-clef (`a` corresponds to the A of nominal frequency 222.5 Hz; the G of the G-clef is denoted `g`). Lower pitch notes are specified using upper case letters ranging from `A` to `N` (the F of the F-clef is denoted `M`, and `F` is one octave below).

[6]But: provided you keep it *not scalable* if you use the `\hardnotes` or `\hardlyrics` features (see 2.20.7).

If necessary, a numeric symbol can be used to place a symbol independently of the active clef, which has the drawback not to be transposable.

Besides, notes below `A` (i.e. the A of nominal frequency 55.625 Hz), namely the lowest octave of the modern pianos, can only be coded using the transposition features (see below: *transposition* and *octaviation*) or in absolute vertical position using numbers.

2.5 Writing notes

There are two major kinds of note macros:

1. those which terminate a note/chord stem and are followed by a horizontal spacing of value `\noteskip`,
2. those which initiate or extend a note/chord stem and do not cause horizontal spacing.

The first kind is used to type a melody, the second kind is used to type chords.

2.5.1 Single spacing notes

`\breve` p : breve note () at pitch p.

`\longa` p : longa note () at pitch p.

`\zmaxima` p : maxima note () at pitch p.

`\wq` p : arbitrary duration note () at pitch p.

`\wh` p : whole note at pitch p.

`\hu` p : half note at pitch p with stem up.

`\hl` p : half note at pitch p with stem down.

`\ha` p : half note at pitch p with automatic stem[7] choice.

`\qu` p : quarter note at pitch p with stem up.

`\ql` p : quarter note at pitch p with stem down.

`\qa` p : quarter note at pitch p with "automatic" stem.

`\cu` p : eighth note[8] at pitch p with stem up.

`\cl` p : eighth note at pitch p with stem down.

`\ca` p : eighth note at pitch p with "automatic" stem.

`\ccu` p : sixteenth note at pitch p with stem up.

`\ccl` p : sixteenth note at pitch p with stem down.

`\cca` p : sixteenth note at pitch p with "automatic" stem.

`\cccu` p : 32-th note at pitch p with stem up.

`\cccl` p : 32-th note at pitch p with stem down.

`\ccca` p : 32-th note at pitch p with "automatic" stem.

`\ccccu` p : 64-th note at pitch p with stem up.

`\ccccl` p : 64-th note at pitch p with stem down.

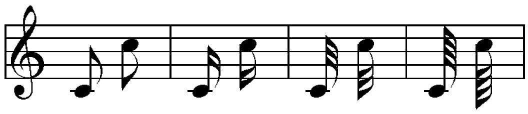
---

[7] This means all stems belonging to notes below the third line are stemmed up, the other down, they work with every clef, but only for single notes.

[8] The `\c` of this macro name is taken from the French word "croche" which is by the way one half of the english "crotchet"; `\cc...`, `\ccc...` are standing for "double croche", "triple croche", etc.

`\ccccl` p : 64-th note at pitch p with “automatic” stem.

As an example, the sequence:



was coded as:

```
\Notes\cu c\cl j\enotes\bar
\Notes\ccu c\ccl j\enotes\bar
\Notes\cccu c\cccl j\enotes\bar
\Notes\ccccu c\ccccl j\enotes
```

If these notes are preceded by *non-spacing* notes (i.e. macros `\zq` or `\zh`) their stem is extended up or down so as to join all notes into a single chord.

2.5.2 Non-spacing (chord) notes

`\zq` p : quarter (or shorter) note head at pitch p with no spacing after.

`\zh` p : half note head at pitch p with no spacing after.

It must be pointed out that the pitch p of these notes is memorized so that the stem of the further spacing note will join them into a chord. This stem top and bottom pitch is *reset* at each spacing note.

REMARK: Notes of duration longer than whole notes are always non-spacing. This saves one useless definition, since these notes are always longer than other simultaneous ones. If needed they can be followed by `\sk` *to force spacing.*

2.5.3 Shifted non-spacing (chord) heads

These symbols are used mainly in chords where *second* intervals are present. It is the responsibility of the typist to choose which heads should be shifted left or right.

`\rw` p : whole note head shifted right by one note width ($\approx$ 6pt), no spacing.

`\lw` p : whole note head shifted left by one note width ($\approx$ 6pt), no spacing.

`\rh` p : half note head shifted right by one note width ($\approx$ 6pt), no spacing.

`\lh` p : half note head shifted left by one note width ($\approx$ 6pt), no spacing.

`\rq` p : quarter note head shifted right by one note width ($\approx$ 6pt), no spacing.

`\lq` p : quarter note head shifted left by one note width ($\approx$ 6pt), no spacing.

Except that they are shifted left of right, these macros act like `\z...` macros for stem building.

2.5.4 Single non-spacing notes

`\zhu` : half note with stem up but no spacing. It acts like `\hu` for stem building.

`\zhl` : half note with stem down but no spacing. It acts like `\hl` for stem building.

`\zqu` : quarter note with stem up but no spacing. It acts like `\qu` for stem building.

`\zql` : quarter note with stem down but no spacing. It acts like `\ql` for stem building.

`\zcu`, `\zccu`, `\zcccu`, `\zccccu` : eighth, ..., note with stem up but no spacing. It acts like `\cu` for stem building.

`\zcl`, `\zccl`, `\zcccl`, `\zccccl` : eighth, ..., note with stem down but no spacing. It acts like `\cl` for stem building.

`\zqb` : note belonging to a beam but no spacing.

`\rhu`, `\rhl`, `\rqu`, `\rql`, `\rcu`, `\rcl` : `\rhu` acts like `\zhu`, but the note is shifted one note width on the right, other analogous.

`\lhu`, `\lhl`, `\lqu`, `\lql`, `\lcu`, `\lcl` : same as above, but the whole of the note is shifted one note width on the left.

`\zw` *p* : whole note at pitch *p* with no spacing after.

`\zwq` *p* : arbitrary duration note () at pitch *p* with no spacing after.

`\zbreve` *p* : breve note () at pitch *p* with no spacing after.

`\zlonga` *p* : longa note () at pitch *p* with no spacing after.

`\zmaxima` *p* : maxima note () at pitch *p* with no spacing after.

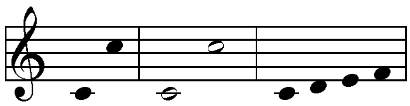
2.5.5 Single (spacing) stemless notes

Although not standard in real music scores, one may need to have stemless quarter and half note heads posted in the same way as whole notes. This can be done with the following commands:

`\nh` *p* : half note head at pitch *p*.

`\nq` *p* : quarter note head at pitch *p*.

As an example, the sequence:



was coded as:

```
\notes\nq c\nq j\enotes\barre
\Notes\nh c\nh j\enotes\barre
\notes\nq {cdef}\enotes
```

In case of special need, non spacing variants have been provided, namely `\znh` and `\znq`.

2.5.6 Pointed notes

One simple way of doing consists in putting `\pt` *p* to get a *dot* after the normal note head at pitch *p*. Thus a quarter note with one point can be coded `\pt h\qu h`, with two points as `\ppt\qu h` and with three points as `\pppt\qu h`.

A simpler way of doing consists in using compact macros, namely: `\whp`, `\whpp`, `\zwp`, `\zwpp`, `\hup`, `\hupp`, `\hlp`, `\hlpp`, `\zhp`, `\zhpp`, `\qup`, `\qupp`, `\qlp`, `\qlpp`, `\zqp`, `\zqpp`, `\cup`, `\cupp`, `\clp`, `\clpp`, `\qbp` and `\qbpp`. Where all `\z...p` are useful in chords.

You may also introduce pointed notes, especially in groups by coding a *period* before (not after) the letter representing the pitch: `\qu{.a.^b.c}` which is equivalent to:

```
\pt{a}\qu{a}\pt{b}\sh{b}\qu{b}\pt{c}\qu{c}
```

Finally, pointed notes can also be produced without spacing after, using `\zhup`, `\zhlp`, `\zqup`, `\zqlp`, `\zcup`, `\zclp`, `\zqbp`, and the same with two `p`'s for double-pointed notes, like `\zhupp`, `\zhlpp`, `\zqupp`, `\zqlpp`, `\zcupp`, `\zclpp` and `\zqbpp`.

If two voices share one staff, the points of the lower voice are lowered, if the note is on a note line. Therefore you can use `\lpt` p and `\lppt` p.

2.6 Beams

Beams are not automatically handled, but they must be declared explicitely, *before* the first spacing note involving them is coded. Two kinds of macros are provided:

1. fixed slope beams have an arbitrary slope chosen by the user in the range -45% to +45% (by multiples of 5%);
2. semi-automatic beams have their slope computed knowing the number of `\noteskip` over which they are supposed to extend, and knowing the initial and final pitch of the notes they are supposed to link.

2.6.1 Fixed slope beams

2.6.1.1 Beam initiation

`\ibu` nps : initiates an *upper beam* 3 horizontal line spacings above the pitch p ; n is its reference number, which must be in the range [0-5] ([0-8] if `musixadd` file has been `\input`); s is the slope of the beam.

s is an integer in the range [-9,9]. $s = 1$ means a slope of 5%, $s = 9$ means a slope of 45%, $s = -3$ means a slope of -15%, etc. With usual spacings a slope of 2 or 3 is fit for ascending scales. A slope of 6 to 9 is fit for ascending arpeggios.

`\ibl` nps : initiates a *lower beam* 3 horizontal line spacings below the pitch p. Other parameters as above.

`\ibbu` nps : initiates a *double upper beam* (same parameter meaning).

`\ibbl` nps : initiates a *double lower beam* (same parameter meaning).

`\ibbbu` nps : initiates a *triple upper beam* (same parameter meaning).

`\ibbbl` nps : initiates a *triple lower beam* (same parameter meaning).

`\ibbbbu` nps : initiates a *quadruple upper beam* (same parameter meaning).

`\ibbbbl` nps : initiates a *quadruple lower beam* (same parameter meaning).

Notes belonging to beams are coded in the form `\qb` np where n is the beam number and p the pitch of the note head. MusiXTEX adjusts the length of the note stem to link the bottom of the chord to a beam.

2.6.1.2 Beam termination

Beam termination is also not automatic. The termination of a given beam must be explicitely declared *before* coding the last spacing note connected to that beam.

`\tbu` n : terminates upper beam number n at current position.

`\tbl` n : terminates lower beam number n at current position.

`\tbu` and `\tbl` terminate beams of any multiplicity. Therefore 32-th notes hanging on a triple beam are initiated by `\ibbbu` *nps* and terminated by `\tbu` *n*.

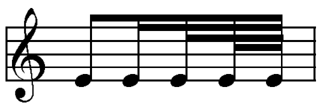
Since beams usualy finish with a `\qb` to link the last note, shorter macros have been provided:

- `\tqb` *np* is equivalent to `\tbl` *n*`\qb` *np*
- `\tqh` *np* is equivalent to `\tbu` *n*`\qb` *np*
- `\ztqb` *np* is equivalent to `\tbl` *n*`\zqb` *np* (no spacing after)
- `\ztqh` *np* is equivalent to `\tbu` *n*`\zqb` *np* (no spacing after)

2.6.1.3 Beams of increasing multiplicity

It is also possible to code beams whose multiplicity is not the same at the beginning. The multiplicity can be increased at any position. For instance, `\nbbu` *n* which sets the multiplicity of upper beam number *n* to 2 starting at the current position, `\nbbbu` *n* sets its multiplicity to 3 and `\nbbbbu` *n* sets its multiplicity to 4. `\nbbl` *n* ...`\nbbbbl` *n* perform the same functions for lower beams.

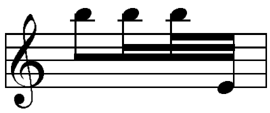
Note that the difference between upper and lower beams does not mainly consist in the beam being above or below the note heads; rather, it specifies whether the abscissa of the beginning and the end of this beam is aligned on the right (upper beam) or on the left (lower) beam. Thus, the sequence:



has been coded as

```
\Notes\ibu0h0\qb0e\nbbu0\qb0e\nbbbu0\qb0e\nbbbbu0\qb0e\tbu0\qb0e\enotes
```

It is quite possible to terminate with `\tbu` a beam initiated with `\ibl`. This may give:

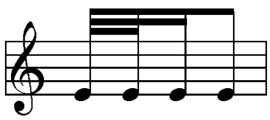


which has been coded as

```
\Notes\ibl0p0\qb0p\nbbl0\qb0p\nbbbl0\qb0p\tbu0\qb0e\enotes
```

2.6.1.4 Beams of decreasing multiplicity

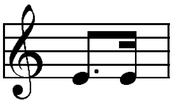
Partial termination of beams is also possible, by using `\tbbu` or `\tbbl` : these macros terminate the current beam except that of order 1 (eighths). `\tbbbu` or `\tbbbl` terminate the current beam except those of order 1 and 2, etc.

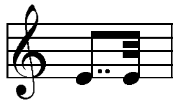


has been coded as

```
\startextract
\Notes\ibbbu0h0\qb0e\tbbbu0\qb0e\tbbu0\qb0e\tbu0\qb0e\enotes
\endextract
```

The macros `\tbbu` and `\tbbl` and higher order may also be invoked when only a single beam is active. Then, a second beam or third or ... (upper or lower according the initiating procedure) is opened *one note width before the current position, and closed immediately*. Thus the following sequences



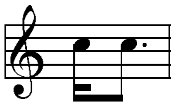


are coded:

```
\Notes\ibu0e0\qbp0e%
  \tbbu0\tbu0\qb0e\en
```

```
\Notes\ibu0e0\qbpp0e%
  \tbbbu0\tbbu0\tbu0\qb0e\en
```

The same behaviour occurs in the case of `\tbbbu`, `\tbbbl`, `\tbbbbu` and `\tbbbbl`. The symmetrical pattern is also possible. For example:



has been coded as:

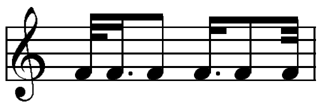
```
\Notes\ibbl0j0\roff{\tbbl0}\qb0j\tbl0\qbp0j\enotes
```

REMARK: these codings may seem complicated. In fact, it is the responsibility of the user to define macros which perform the most common sequences in his masterpiece. For example, one could define sets of four sixteenths by the macro:

```
\def\qqh#1#2#3#4#5{\ibbl0#2#1\qb#2\qb#3\qb#4\tbl0\qb#5}
```

where the first argument is the slope and the other four arguments are the pitches of the four consecutive sixteenths wanted.

A slightly more complicated example is:

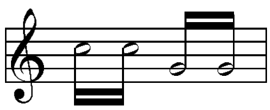


has been coded as:

```
\notes\ibbbu0e0\roff{\tbbbu0}\qb0f\en
\notesp\tbbu0\qbp0f\en
\Notes\tbu0\qb0f\en
\notesp\ibbu0f0\roff{\tbbu0}\qbp0f\en
\Notes\qb0f\en
\notes\tbbbu0\tbbu0\tbu0\qb0f\en
```

2.6.2 Repeated pattern beams

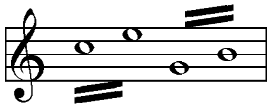
Note heads hanging on beams are not necessarily quarter (or higher order) note heads. It is possible to hang half note heads on beams using `\hb` macro, e.g.:



has been coded as:

```
\Notes\ibbl0j0\hb0j\tbl0\hb0j\enotes
\Notes\ibbu0g0\hb0g\tbu0\hb0g\enotes
```

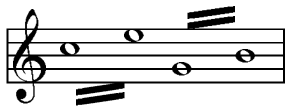
It is also possible to write



which was coded as:

```
\Notes\ibbl0j3\wh j\tbl0\wh l\enotes
\Notes\ibbu0g3\wh g\tbu0\wh i\enotes
```

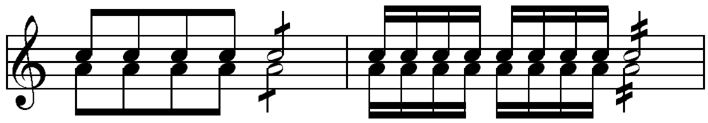
However, a better look could be obtained in a more sophisticated way[9]:



which was simply coded as:

```
\Notes\loff{\zw j}\ibbl0j3\sk\tbl0\wh l\enotes
\Notes\ibbu0g3\wh g\tbu0\roff{\wh i}\enotes\qspace
```

Single repeated notes can also be made — in a much more sophisticated way[10], for example:



whose coding (due to W. Icking) is;

```
\Notes\ibl0h0\qb0{hhh}\tbl0\qb0h\bsk\bsk\bsk\bsk
      \ibu0j0\qb0{jjj}\tbu0\qb0j\en
\NOTes\loffset{0.5}{\ibl0j9}\roffset{0.5}{\tbl0}\zhl h%
      \loffset{0.5}{\ibu0g9}\roffset{0.5}{\tbu0}\hu j\en\bar
\notes\ibbl0i0\qb0{hhh}\tbl0\qb0h\bsk\bsk\bsk\bsk
      \ibbu0i0\qb0{jjj}\tbu0\qb0j%
      \ibbl0i0\qb0{hhh}\tbl0\qb0h\bsk\bsk\bsk\bsk
      \ibbu0i0\qb0{jjj}\tbu0\qb0j\en
\NOTes\loffset{0.5}{\ibbl0k9}\roffset{0.5}{\tbl0}\zhl h%
      \loffset{0.5}{\ibbu0f9}\roffset{0.5}{\tbu0}\hu j\en
```

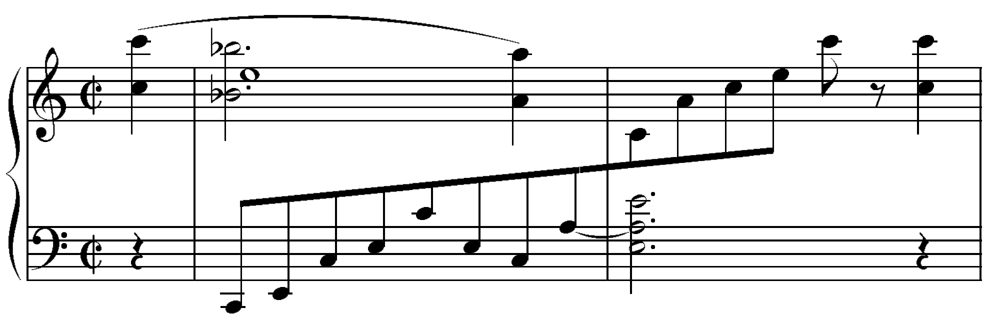
2.6.3 Beams across bars

The usual way most composers wrote their scores was beams inside bars, at least until the beginning of the XIX-th century. Unfortunately, later composers (Brahms, Scriabin, Grieg, etc.) wanted to write beams jumping across bars. This is possible without any problems. We give an example from Brahms's Intermezzo op. 118,1 provided by Miguel Filgueiras:

---

[9] You are suggested to make your own macro if you have to type many of these, or better: use a pre-compiler.

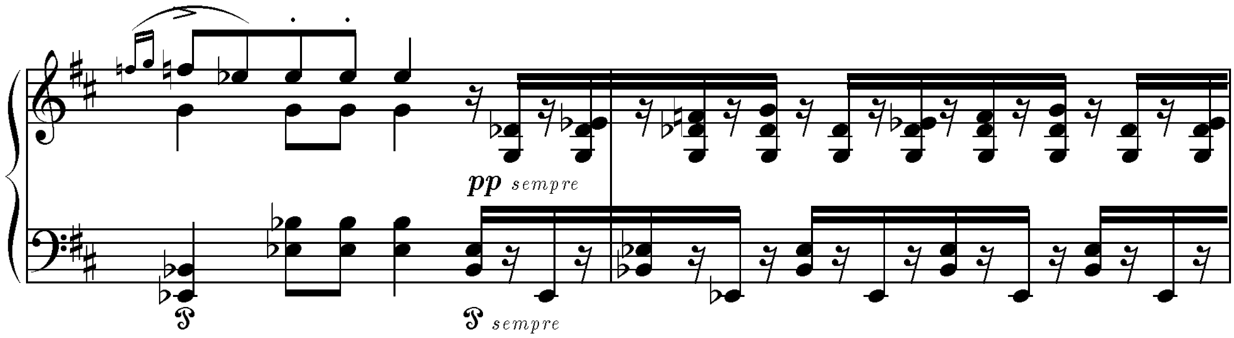
[10] But you can write your own macro, thinking of using `\transpose` to lift the oblique beams.



whose coding is:

```
\interstaff{13}
\instrumentnumber{1}
\setstaffs1{2}
\setclef1\bass
\generalmeter\allabreve
\startextract
\NOtes\qp\nextstaff\isluru0q\zq{q}\ql{j}\enotes
\bar
\nspace
\Notes\ibu0a1\qb0{CEJLcL}%
  \nextstaff\roff{\zw{l}}\pt{p}\zh{_p}\pt{i}\hl{_i}\enotes
\Notes\qb0J\itied1a\qb0a\nextstaff\tslur0o\zq{o}\ql{h}\enotes
\bar
\Notes\ttie1\zh{.L.a}\hl{.e}%
  \nextstaff\qb0{chj}\tbl0\qb0l\cl{q}\ds\enotes
\NOtes\qp\nextstaff\zq{q}\ql{j}\enotes
\endextract
```

Still to be done manually is the cut and prolongation across line breaks, which can be easily be done with shifting using `\roff` and/or `\loff`, or with insert of a spacing command (here done with `\hsk`). We give an example from Grieg's "Hochzeit auf Troldhaugen":



where the prolongation was coded as:

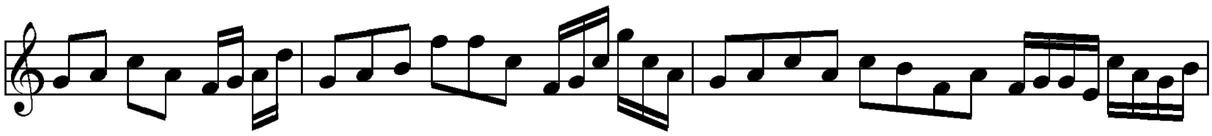
```
\notes\rlap{\qs}\hsk\tbu0|\rq e\zq d\zqb1N\hsk\tbu1\en
```

2.6.4 Semi-automatic beams

In order to avoid tedious checks to adjust the slope (and even the starting pitch) of beams in music with a lot of steep beams, a set of automatically slope computing has recently been implemented. If you say `\Ibu2gj3` MusiXTEX will understand that‘you want to build an upper beam (beam number 2) horizontally extending `3\noteskip`, the first note of which is a `g` and the last note is a `j`. Knowing these parameters it will choose the highest slope number which corresponds to a slope not more than (`j` − `g`)/(3`\noteskip`). Moreover, if there is no sufficiently steep beam slope available, then it will raise the starting point.

Eight such macros are available: `\Ibu`, `\Ibbu`, `\Ibbbu`, `\Ibbbbu`, `\Ibl`, `\Ibbl`, `\Ibbbl` and `\Ibbbbl`.

Also available are ready definitions for often needed sets of double, triple and quadruple notes with computed slopes. These are: `\Dqbu`, `\Dqbl`, `\Dqbbu`, `\Dqbbl`, `\Tqbu`, `\Tqbl`, `\Tqbbu`, `\Tqbbl`, `\Qqbu`, `\Qqbl`, `\Qqbbu` and `\Qqbbl`.



which was simply coded as:

```
\Notes\Dqbu gh\Dqbl jh\en
\notes\Dqbbu fg\Dqbbl hk\en\bar
\Notes\Tqbu ghi\Tqbl mmj\en
\notes\Tqbbu fgj\Tqbbl njh\en\bar
\Notes\Qqbu ghjh\Qqbl jifh\en
\notes\Qqbbu fgge\Qqbbl jhgi\en
```

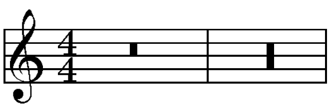
2.7 Rests

Except that difference that they have no specific pitch, rests are coded in a very simple way.

2.7.1 Ordinary rests

Full bar rests (also called "pauses") are coded as `\pause`, with a point behind as `\pausep`; smaller rests are `\hpause` (of duration equal to a *half note*), with a point behind as `\hpausep`, `\qp` (duration equal to a *quarter note*, also `\soupir`), `\ds` (duration equal to an *eighth note*), `\qs` (duration equal to an *sixteenth note*), `\hs` (duration equal to an *thirty second note*), `\qqs` (duration equal to an *sixty fourth note*).

Long rests (lasting several bars) can be coded as `\PAuse` and `\PAUSe`, which respectively yield:



2.7.2 Lifted rests

All the previous rests with exception of `\pausep` (pause with a point) and `\hpausep` (half rest with a point) are *hboxes*; this means that they can be vertically offset to meet polyphonic music requirements using the standard TEX command `\raise`. For example:

```
\raise 2\Interligne\qp
\raise 3mm\qq
```

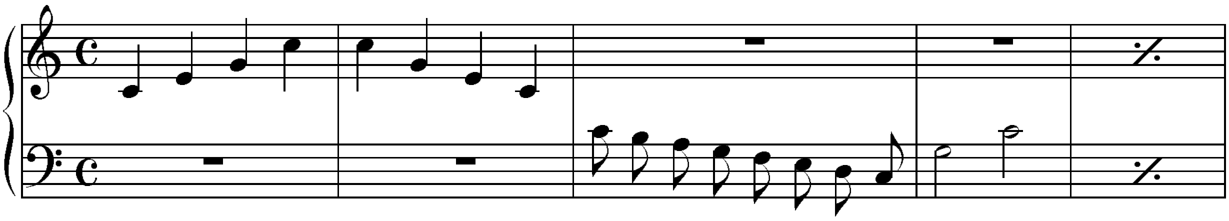
CAUTION: *do not try to lift rests (or notes but this looks so silly that...) using codings like* `\raise2mm\hbox{\qp}`*. The reason is that the rest symbol (e.g.* `\qp`*) is embedded in braces, so that local counting of spaces (register* `\loc@skip`*) will be lost, resulting in turn in overfull/undfull boxes, beam errors, erratic slurring, etc. In general, all rests and all notes should be coded at the brace level of the* `\notes` *command, unless they are deliberately embedded in* `\rlap` *or* `\llap` *which actually overlook spacing counting but finally force no horizontal spacing, a behaviour which compensates the forgotten spacing count.*

In addition, two symbols have been provided to put a *full rest* or a *half rest* above or below the staff. Then the ordinary `\pause` or `\hpause` cannot be used since there is a need for small horizontal line to distinguish between the full and the half rest. They are:

- `\liftpause` n (non spacing) to get a raised from original position by n staff line intervals,
- `\lifthpause` n (non spacing) to get raised the same way.
- `\liftpausep` n (non spacing) to get a raised from original position by n staff line intervals,
- `\lifthpausep` n (non spacing) to get raised the same way.

2.7.3 Bar centered rests

Sometimes it is necessary to place a rest (or other) exactly in the middle of a bar. This can be done with more sophisticated commands: `\centerbar`, `\centerPAUSe`, `\centerPAuse`, `\centerpause`, `\centerhpause`. For example:

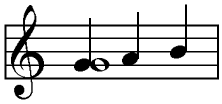


```
\generalmeter\meterC
\setclef1\bass
\setstaffs1{2}
\parindent0pt
\startpiece\addspace\afterruleskip
\NOtes|\qa{cegj}\en
\def\atnextbar{\znotes\centerpause\en}\bar
\NOtes|\qa{jgec}\en
\def\atnextbar{\znotes\centerpause\en}\bar
\Notes\ca{`jihgfedc}\en
\def\atnextbar{\znotes|\centerpause\en}\bar
\NOTes\ha{Nc}\en
\def\atnextbar{\znotes|\centerpause\en}\bar
\addspace{10\elemskip}%
\def\atnextbar{\znotes\centerbar{\duevolte}|\centerbar{\duevolte}\en}\endpiece
```

2.8 Phantom notes and spacing commands

It may be interesting, when coding a sequence of notes within a unique pair `\notes...\enotes`, to skip one note place in order – for example – to set the third note of one staff at the same abscissa as that of the second note of another staff. This can be done by inserting `\sk` which causes a spacing of one `\noteskip`[11]. Sometimes it is useful to *skip back* one noteskip, this can be done with `\bsk`. The spacing of a half `\noteskip` can be yielded with `\hsk`.

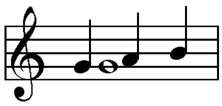
If you just want to shift a note or a symbol by one note head width, you may write `\roff{`*note/symbol*`}` (shift right) or `\loff{`*note/symbol*`}` (shift left). If you just want to shift a note or a symbol by one half note head width, you may write `\hroff{`*note/symbol*`}` (shift right) or `\hloff{`*note/symbol*`}` (shift left). For example, to get



you code:

```
\Notes\roff{\zwh g}\qu g\qu h\qu i\enotes
```

If you want to shift notes or symbols right by more or less than one note head width, then you can use `\roffset` (or `\loffset` in the other direction) which has an additional first argument which is the number of note head widths the second argument should be shifted by. For example



was coded:

```
\Notes\roffset{1.5}{\zwh g}\qu g\qu h\qu i\enotes
```

IMPORTANT: the offset specified or implicitly specified in `\roff`*,* `\loff`*,* `\roffset` *and* `loffset` *does not add to the total spacing amount; in other words this is just an offset, not a spacing. Conversely, the possible spacing of commands included in the argument of* `\roff/\loff` *and in the second argument of* `\roffset/\loffset` *actually adds to the total spacing in that bar of the current staff. Therefore, the symbol/note argument of these offset commands should usually produce no spacing, i.e. begin with* `\z`.

Besides, it is not advised to insert several consecutive spacing notes in the `\roff/\loff` *or* `\roffset/\loffset` *arguments, since it could mess up the global spacing accounting.*

To insert spacing of *nearly* one *note head width*, you can use `\qsk`(which has the drawback, that it is scalable, so what spacing you really get depends on the computed `\elemskip` from `musixflx` and the user chosen value of `\elemskip`). The half of this spacing can be forced with `\hqsk`. Note that these two latter macros must be used inside a pair `\notes...\enotes`.

If you want to insert additional spacing before a group of notes, especially to avoid a collision with an accidental, you can write `\nspace` — outside the `\notes...\enotes` group — and this will produces an additional spacing of a half note head width; in the same way, `\qspace` will yield a spacing of a note head width. Note that `\nspace` and `\qspace` are "hard" spacings, whose general feature is `\hardspace`, described below.

[11]Never use `\kern` nor `\hskip` : in fact `\sk` not only causes a space but also records that space for correct handling of beams, slurs, trills, octaviation dashed lines, line-breaking, etc.

Sometimes the additional space behind `\changecontext` disturbs the eye of a typesetter, but this ugly spacing can be reduced with:

`\addspace{-\afterruleskip}%`

It should be emphasized that all these spacing commands work correctly, only when their dimension arguments are *scalable values* and not fixed dimensions; the scalable dimensions are `\elemskip`, `\beforeruleskip`, `\afterruleskip`, `\noteskip` and their multiples.

The only spacing command which can work with "normal" (that is, *not scalable*) dimensions is:

`\hardspace{`*any TEX dimension*`}%`

but both `\addspace` and `\hardspace` can only be used outside of `\notes...\enotes`.

Conversely, a more general spacing is allowed within the `\notes...\enotes` pairs, namely:

`\off{`D`}%`

where D is a *scalable dimension*, for example `\noteskip` or `\elemskip`. In fact, if you look to the MusiXTEX source, you will find that `\off` is the basic control sequence used to define `\sk`, `\qsk`, etc.

2.9 Collective coding: sequences of notes

As seen in the MOZART example, it is not necessary to write a macro sequence `\notes...\enotes` for each column[12]. If, on all staffs of all instruments, spacings are equal or multiple of a unique value, the notes may be concatenated in each staff: each note in each staff makes the current position horizontally advance by the elementary spacing specified by the choice of `\notes`, `\Notes`, `\NOtes`, etc.

The major interest of this feature resides in that fact that the note macros are able to write several items; for instance `\qu{cdefghij}` writes the *C-major* scale in quarters with stem up. In the same way `\cl{abcdef^gh}` writes the *A-minor* scale in eighths. Not all note generating macros can be used to perform collective coding, but most of them can.

If necessary a void space[13] can be inserted in a collective coding by using `*`.

2.10 Accidentals

Accidentals can be introduced in two ways.

The first way, the *manual* way of coding them, consists for example in coding `\fla` to put a *flat* at the pitch a, supposedly before the further note of that pitch. There is no control upon the fact that a note will be put at this position and at this pitch. Naturals, sharps, double flats and double sharps are coded `\na` p, `\sh` p, `\dfl` p and `\dsh` p respectively.

Alternate procedures `\lfl`, `\lna`, `\lsh`, `\ldfl` and `\ldsh` place the same accidentals, but their abscissa is shifted one note head width on the left. The purpose of this is to avoid collision of accidentals in a chord with narrow intervals.

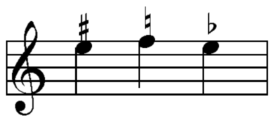
The second way of coding accidentals consists in putting the symbol `^` (sharp), the symbol `_` (flat), the symbol `=` (natural), the symbol `>` (double sharp), or the symbol `<` (double flat) within the coding of the note, e.g.: `\qb{^g}` yields a $G\sharp$. This may very well be combined with collective coding, e.g.: `\qu{ac^d}`.

[12] Although the compiler compiles it faster.

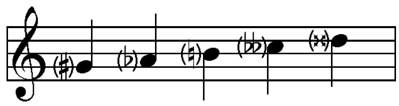
[13] same behaviour as `\sk`

Two sizes are available for accidentals. They revert to the small version when notes are supposed to be too close to each other. These two sizes can be forced by coding \bigfl, \bigsh, etc., or \smallfl, \smallsh, etc. If one does not want to have any small accidentals, then one can declare \bigaccid (conversely \smallaccid or \varaccid – the latter restoring variable sizes).

Small accidentals can also be put *above* the note heads. This is done using \uppersh *p*, \upperna *p* or \upperfl *p*:



It also possible to introduce *cautionary accidentals* on a score, i.e. small size accidentals between parentheses. This done by preceding the name of the accidental keyword by a c, e.g. by coding \cfl *p* to get a cautionary flat. Available cautionary accidentals are \csh, \cfl, \cna, \cdfl and \cdsh, which give:



Besides, the distance between note and accidental is influenced by

\accshift=*any TEX dimension*

positive values shift to left, negative to right, default is 0pt.

2.11 Transposition and octaviation

An important feature is the existence of a special register \transpose the normal value of which is 0. If you say

\transpose=3

all subsequent pitches specified by upper or lower case letters[14] will be transposed 3 positions. If you set \transpose to 7 you may write your music one octave below its final pitch. Thus, you can define *octaviation* macros like

\def\soqu#1{\zq{#1}{\transpose=7 \qu{#1}}}

to build quarter note octaves in a single call. Note that the octaviated note is coded within braces so that the transposition is only local.

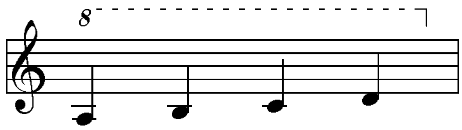
Octaviation can also be performed in another way, namely using special codes to transpose by multiples of 7 intervals. For example \qu{'ab} is equivalent to \qu{hi} and \qu{`kl} is equivalent to \qu{de}. It should be emphasized here that the ' (*acute accent*) and the ` (*grave accent*) have cumulative effects, so that \qu{''A'A} is equivalent to \qu{ah} and that the \transpose parameter is only reset to its initial value (not necessarily zero) when changing staff or instrument (i.e. | or &) or at \enotes. Since this may be confusing, it is useful to use the ! prefix to reset the \transpose register explicitely to the value it had when entering \notes[15]. Thus \qu{!a'a} always gives the note a and its upper octave h *shifted by the value of* \transpose *at the beginning of the current* \notes...\enotes group (or \Notes...\enotes, etc.) whatever the number of previous grave and acute accents occurring inbetween.

[14] Pitches specified with figures are absolute and not transposed.

[15] This value is saved in another register named \normaltranspose.

2.11.1 Typical piano octave transposition

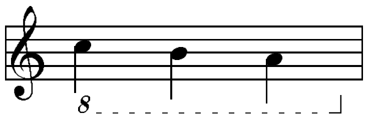
2.11.1.1 Local octave transposition of fixed length



can be coded as

```
\startextract
\NOTEs\octfinup{10}{3.5}\qu a\qu b\qu c\qu d\en
\endextract
```

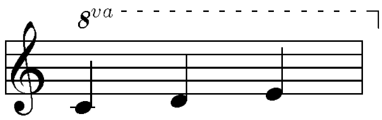
that is, the dashed line extends 3.5×`\noteskip`. Conversely, lower octaviation can also be coded, for example:



which is coded as

```
\startextract
\NOTEs\octfindown{-5}{2.6}\ql j\ql i\ql h\en
\endextract
```

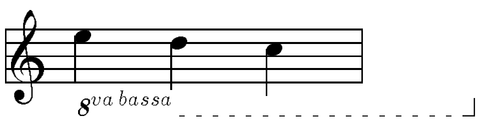
If a more sophisticated posting is wanted, this can be achieved refefining the macro `\octnumber`, e.g.:



whose coding is

```
\startextract
\NOTEs\def\octnumber{\ppffsixteen8$^{va}$}\octfinup{10}3\qu c\qu d\qu e\en
\endextract
```

or



coded as:

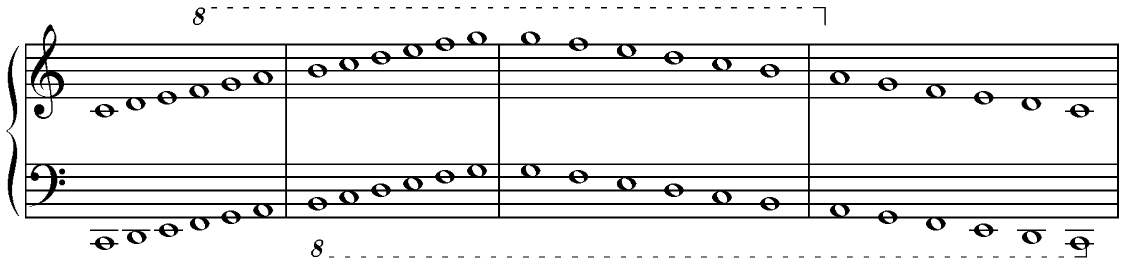
```
\startextract
\NOTEs\def\octnumber{\ppffsixteen8$^{va\,bassa}$}%
  \octfindown{-5}4\ql l\ql k\ql j\en
\endextract
```

2.11.1.2 Long or variable range octave transposition

Long range octave transpositions are better handled using `\Ioctfinup`, `\Ioctfindown` and `\Toctfin` whose effect extends over line breaks.

`\Ioctfinup` *np* starts an octave transposition of reference number n (with $0 \leq n < 6 \equiv$`\maxoctlines`) dashed line at pitch p (usually $p > 10$ but it can also be an alphabetic pitch specification), `\Ioctfindown` np starts a lower octave transposition at pitch p (usually $p < -2$), and both extend until terminated with `\toctfin`. The difference between `\Ioctfinup` n and `\Ioctfindown` n is the relative position of the "8" figure with respect to the dashed line, and the sense of the terminating hook (note that `\Ioctfinup` has a hook under the dashed line, and conversely).

For the sake of backward compatibility, `\ioctfinup` is equivalent to `\Ioctfinup 0` and `\ioctfindown` is equivalent to `\Ioctfindown 0`. For example:



whose coding is

```
\begin{music}
\instrumentnumber{1}
\setstaffs12
\setclef1{6000}
%
\startextract
\notes\wh{CDEFGH}|\wh{cde}\Ioctfinup 1p\wh{fgh}\enotes
\bar
\notes\Ioctfindown 2A\wh{IJKLMN}|\wh{ijklmn}\enotes
\bar
\Notes\wh{NMLKJI}|\wh{nmlkji}\Toctfin1\enotes
\bar
\Notes\wh{HGFED}\Toctfin2\wh C|\wh{hgfedc}\enotes
\endextract
\end{music}
```

The elevation of octaviation lines may be raised/lowered using `\Liftoctline` n p where n is the reference number of the wanted ocave line, and n a number (possibly negative) number of `\internote` by which the dashed line should be lifted. This is particularly useful when octaviation lines last several systems and need to be lifted in the systems occuring farther than the initiation.

2.11.2 Transposition of accidentals

The above processes indeed change the vertical position of the note heads and associated symbols (note stems, accents and beams) but they do not take care of the necessary changes of accidentals when transposing, i.e. the fact that an $F\sharp$ occurring with a zero signature should become a $B\natural$ when transposing from the tonality of C major to F major where the normal B is the $B\flat$. Since the intent of the composer is not obvious – he may want to shift a group of notes within the same tonality or conversely to transpose it in another tonality – this is not done automatically. Thus the `\sh`, `\fl`, `\na`, `\dsh` and `\dfl` symbols *are not affected* by a change of the `\transpose` register.

But the composer/typesetter may ask MusiXTEX to do that work. In this case, he should code `\relativeaccid`. In that case, a `\sh` command means that the corresponding pitch[16] has to be raised by *one half pitch* with respect to its normal value *according to the current signature*. Thus `\sh b` (using `\relativeaccid`) means a $B\sharp$ if the signature is zero or positive, and a $B\natural$ if it is negative. The same logic applies for all accidentals using `\relativeaccid`.

In the same way, the compact codes `^`, `_`, `=` are normally not affected by transposition and signatures, but their behaviour will be changed by saying `\relativeaccid` and reset by `\absoluteaccid` (the default situation)[17].

Although *relative accidental coding* is an easy and safe way of coding *transposable* scores, care should be exercised in getting rid of the habit of saying `\na b` to raise the pitch of a B when the tonality is F major (i.e. with `\setsign`*n*`=-1` or `\generalsignature{-1}`). An example of sophisticated transposition is given in the score `souvenir.tex` (which `\inputs` `souvenix.tex`).

2.12 Slurs and ties

The slurs and ties provided by MusiXTEX can be divided into two categories:

- Those where the complete slur symbol is composed of a single character from one of the slur fonts, and
- those where the slur symbol is composed three discrete characters, to form the beginning, middle and end of the slur.

The former are called *simple slurs* and the latter are called *compound slurs*. To some extent the division between the two is invisible to the user, in that a number of the macros described below will select between the two types automatically. However, other macros are provided to enable simple slurs to be specified. Note that slurs of both types must be coded within the pair `\notes...\enotes`.

2.12.1 General slur coding

This section describes the usual method of slur coding, where the choice between simple or compound slurs is made automatically. In this case, slurs are initiated and terminated by separate macros, as is the case for beams.

2.12.1.1 Slur initiation

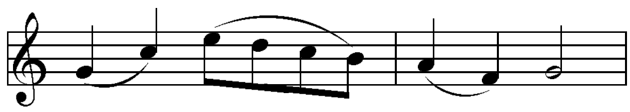
The slur must be initiated *before* the spacing note at which the slur begins, and terminated *before* the note at which the slur ends. The simplest slur initiation macro is

`\isluru` *np*

which initiates an upper slur, with reference number n, beginning at pitch p. The starting point of the slur is centred above an imaginary quarter note head at pitch p. As for beams, the reference number n takes values from 0 to 5, or 0 to 8 if `musixadd.tex` is included. Similarly, `\islurd` *np* initiates a lower slur. These slurs are terminated by coding `\tslur` *np* where n is the reference number and p is the termination pitch. To illustrate with a simple example, the following passage

[16]The musical output note, not the typesetting position.

[17]Note that the behaviour of `\sh`, `\na` and `\fl` with MusiXTEX corresponds to the behaviour of `\Sh`, `\Na` and `\Fl` with MusicTEX+musictrp.tex. But the `musixcpt.tex` file forces the same behaviour as MusiXTEX in that respect, see page 82.



was coded as:

```
\NOtes\islurd0g\qu g\tslur0{'c}\qu c\en
\Notes\isluru0{'e}\ibl0e{-2}\qb0{edc}\tslur0b\tqb0b\en
\bar
\NOtes\islurd0{'a}\qu a\tslur0{`f}\qu f\en
\NOTes\hu g\en
```

Other macros are provided to change the starting and ending point of the slur in relation to the initial and final notes. Thus, `\issluru` *np* initiates a "short" upper slur suitable for linking notes involved in chords. The starting point is shifted to the right, and is vertically aligned with the centre of an imaginary quarter note head at pitch *p*. If a lower short slur is wanted, one should use `\isslurd` *np*.

Sometimes, busy scores call for slurs which are vertically aligned with the ends of note stems rather than note heads. These "beam" slurs — so called because the slur is written at usual beam height — are provided by the macros `\ibsluru` *np* and `\ibslurd` *np*. These macros initiate slurs raised or lowered by the current stem height to accommodate stems or beams above or below.

2.12.1.2 Slur termination

Termination of slurs can be achieved in two ways. First, macros matching each of the initialization macros are provided, as shown below.

Initiation	Termination
`\isluru`, `\islurd`	`\tslur`
`\issluru`, `\isslurd`	`\tsslur`
`\ibsluru`	`\tbsluru`
`\ibslurd`	`\tbslurd`

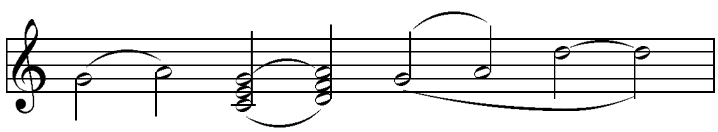
These specific termination macros enable slurs started in one way to be terminated in another. For example, a slur beginning as a "beam" slur may be terminated as a normal slur. This would be achieved using the macro pair `\ibslur...\tslur`.

2.12.1.3 Ties

The coding of *ties* follows from the above discussion as a special case of slurs, i.e. when there is no pitch change. Upper ties are initiated by `\itieu` *np* which starts an upper tie of reference number *n* at pitch *p*. Lower ties are initiated by `\itied` *np* which starts an lower tie of reference number *n* at pitch *p*.

The starting position of the tie is the same as `\issluru` and `\isslurd` respectively. The tie is terminated by coding `\ttie` *n*.

Some simple examples of slurs and ties are illustrated below.



This was coded as:

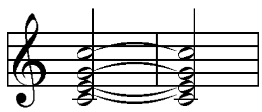
```
\begin{music}
\startextract
\NOTes\isluru0g\hl g\tslur0h\hl h\en
\NOTes\islurd0c\issluru1g\zh{ce}\hu g\tslur0d\tsslur1h\zh{df}\hu h\en
\NOTes\ibsluru0g\islurd1g\hu g\tubslur0h\hu h\en
\NOTes\itieu0k\hl k\ttie0\tdbslur1f\hl k\en
\endextract
\end{music}
```

2.12.1.4 Short ties

Usual music coding makes ties not different of slurs, i.e. a tie is typed exactly like a slur linking two notes of same pitch. However, this traditional way of doing has two the drawback:

- in some special cases, a slur might be confused with a tie and conversely,
- chord ties cannot be implemented with the slur starting above or below the note, but only starting just right of the noetead and finishing just left of the other note head.

This can be done using `\issluru` (2.12.1.2, p. 39) or `\itenu` *np*, `\itenl` *np* for tie initiation, and `\tten` *n* to terminate it. As an example:

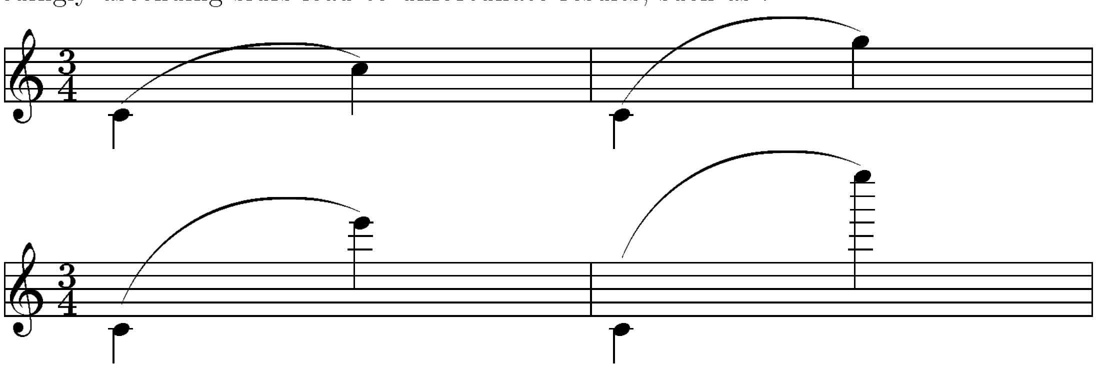


was coded as:

```
\begin{music}
\startextract
\NOTes\itenl0d\itenl1e\itenu2g\itenu3j\zh{ceg}\hu j\enotes
\bar
\NOTes\tten0\tten1\tten2\tten3\zh{ceg}\hu j\enotes
\endextract
\end{music}
```

2.12.1.5 Slur limitations

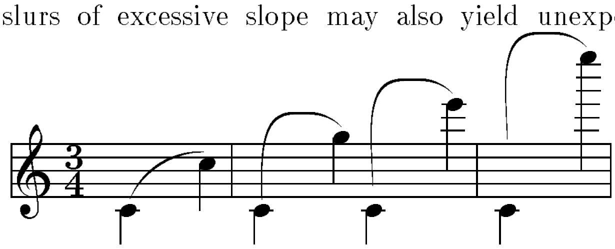
The vertical gap between slur initiation and slur termination is limited to 16`\Internote`. Thus exceedingly ascending slurs lead to unfortunate results, such as :



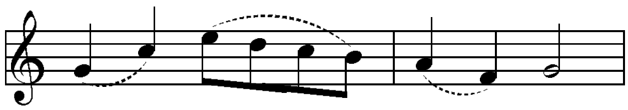
whose coding was:

```
\startextract\NOTes\multnoteskip3\isluru0c\ql c\tslur0j\ql j\enotes
  \bar\NOTes\multnoteskip3\isluru0c\ql c\tslur0n\ql n\enotes\endextract

\startextract\NOTes\multnoteskip3\isluru0c\ql c\tslur0s\ql s\enotes
  \bar\NOTes\multnoteskip3\isluru0c\ql c\tslur0z\ql z\enotes\endextract
 \end{music}
```

Besides, tentative slurs of excessive slope may also yield unexpected shapes, such as:

2.12.2 Dotted slurs

Compound and simple slurs (2.12, p.38) may be drawn dotted[18] specifying `\dotted` just before slur initiation.



This was coded as:

```
\NOtes\dotted\islurd0g\qu g\tslur0{'c}\qu c\en
\Notes\dotted\isluru0{'e}\ibl0e{-2}\qb0{edc}\tslur0b\tqb0b\en\bar
\NOtes\dotted\slur{'a}{`f}d1\qu{'a`f}\en
\NOTes\hu g\en
```

2.12.3 *Modifying slur properties

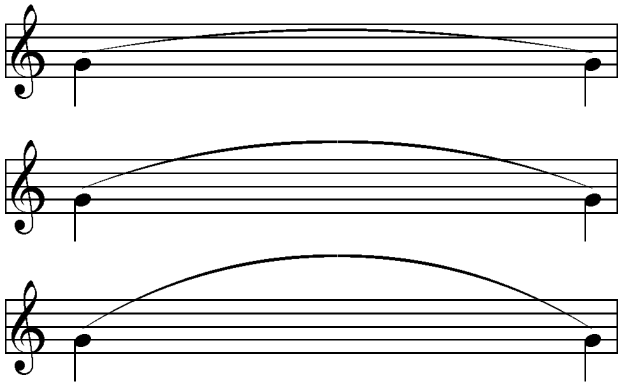
Several macros are provided to modify the shape of slurs already initiated. These macros must be coded before the slur termination. Note that only compound slurs can be modified. Hence, calling any modification macro forces the slur type to be compound.

2.12.3.1 *Changing the rise or fall

By default, the arch of a slur rises and falls from its original height by three times the vertical note spacing. This can be changed using the macro `\midslur` h where h is the revised vertical displacement. For example, `\midslur{6}` coded before `\tslur` causes an upper slur to rise to a maximum height of `6\internote` above the starting position. Note that `\midslur` must be coded *immediately* before the slur termination (eg, `\tslur`). Coding `\midslur` before setting a simple slur causes problems and should be avoided.

---

[18]Thanks to Werner Icking.



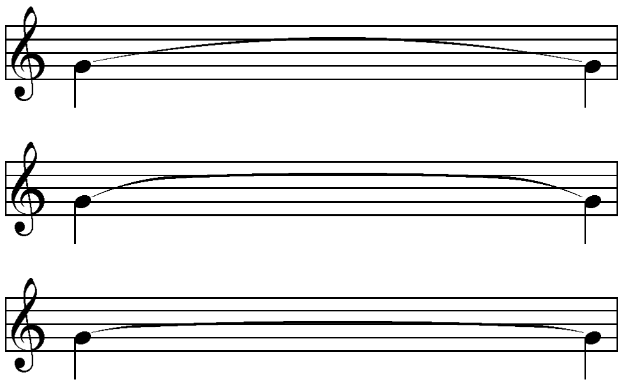
This was coded as:

```
\startextract
\NOtes\multnoteskip8\isluru0g\ql g\en
\notes\tslur0g\ql g\en
\endextract
\startextract
\NOtes\multnoteskip8\isluru0g\ql g\en
\notes\midslur7\tslur0g\ql g\en
\endextract
\startextract
\NOtes\multnoteskip8\isluru2g\ql g\en
\notes\midslur{11}\tslur2g\ql g\en
\endextract
```

2.12.3.2 *Changing the curvature

The degree of curvature depends primarily on the initial and terminal gradient of the slur, relative to its mean slope. The macro `\curve` *hij* allows these to be modified. The first parameter h is the vertical deviation as for `\midslur` described above. The second parameter i sets the initial gradient, while the third parameter j sets the final gradient. The latter parameters are defined as the horizontal distance required to attain maximum vertical deviation. Thus smaller numbers for i and j lead to more extreme gradients. The default setting is `\curve344`. Hence, coding `\curve322` doubles the initial and final gradient relative to the default. As with `\midslur`, `\curve` must be coded *immediately* before the slur termination. Likewise, coding `\curve` before a simple slur causes problems and should be avoided.

The example below illustrates the use of `\curve` more clearly.



This was coded as:

```
\startextract
\NOtes\multnoteskip8\itieu0g\ql g\en
\notes\ttie0\ql g\en
\endextract
\startextract
\NOtes\multnoteskip8\itieu1g\ql g\en
\notes\curve 322\ttie1\ql g\en
\endextract
\startextract
\NOtes\multnoteskip8\itieu2g\ql g\en
\notes\curve 111\ttie2\ql g\en
\endextract
```

2.12.3.3 *Breaking slurs across a line

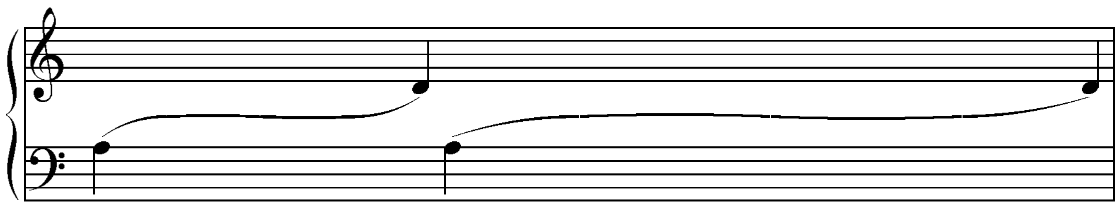
Two macros are provided to control the behaviour of slurs which extend across line breaks. Normally, the part of the slur before the line break is treated as a tie. This can be changed using `\breakslur` *np*, which sets the termination height of the broken slur at the line break to pitch *p*, for slur reference number *n*.

After the line break, the slur is normally resumed at the initial pitch reference, the one coded in `\islur`. To change this, the macro `\Liftslur` *np* may be used. Here *n* is again the slur reference number and *p* is the change in height relative to the initialization height. This macro is normally used following line breaks, in which case it is best coded using the `\atnextline` macro. For example, coding `\def\atnextline{\Liftslur06}` raises the continuation of slur zero by `6\internote` relative to its initialization height.

These macros are illustrated by the following example.

2.12.3.4 *Inverting slur termination

Occasionally in keyboard works one needs to begin a slur in one stave but end it in another. This can be done using the macro `\invertslur` *n* which is best described by reference to the example shown below.



This was coded as:

```
\setstaffs1{2}
\setclef1{\bass}
\startextract
\NOtes\multnoteskip5\isluru0a\ql a\en
\notes\invertslur0\curve311\tslur0g|\qu d\en
\NOtes\multnoteskip{10}\isluru0a\ql a\en
\notes\invertslur0\curve333\tslur0g|\qu d\en
\endextract
```

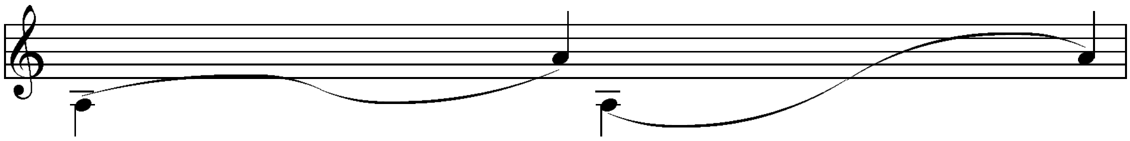
REMARK: Slur inversion is achieved at a place where its slope ise zero ; therefore it only works with ascending slurs started with `\isluru`*, and with descending slurs started with* `\islurd`*. Otherwise no horizontal place can be found and the result is erratic.*

This second situation can be solved manually, stopping the slur at a reasonable place with `\tslur` *and restarting in the other sense at the same place. Stopping and restarting obviously refer to virtual note pitches (a vertical position without actual note) which should be adjusted to have the minimum discontinuity.*

To make this drawing easier, three commands have been provided:

- `\Tslurbreak`*np* stops slur number *n* exactly at pitch *p*, not above or below the virtual note head.
- `\Islurubreak`*np* restarts an upper slur at the same position, not above a virtual note head.
- `\Islurdbreak`*np* restarts an lower slur at the same position, not below a virtual note head.

Thus, the following pattern



was coded as

```
 \begin{music}
\setclef1\treble
\startextract
\NOTes\multnoteskip 3\isluru0a\ql a\en
\NOTes\multnoteskip 3\Tslurbreak0d\Islurdbreak0d\sk\en
\Notes\tslur0h\qu h\en
\NOTes\multnoteskip 3\islurd0a\ql a\en
\NOTes\multnoteskip 3\Tslurbreak0d\Islurubreak0d\sk\en
\Notes\tslur0h\qu h\en
\endextract
\end{music}
```

2.12.4 Simple slurs

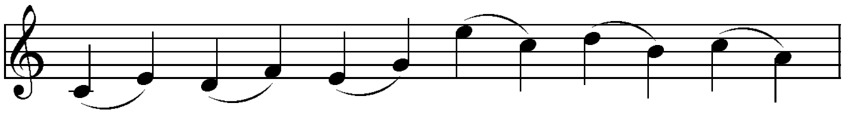
Simple slurs and ties have the advantage of optimal aesthetics and simple coding, but are limited in length to 68pt for slurs and 220pt for ties. Also, the maximum vertical extent of simple slurs is 8 times the internote spacing, and the slurs may not extend across a line break. Despite all these limitations, simple slurs are extremely useful in many applications where the slurs are short and contained within a bar.

Simple slurs extend to the right of the note immediately following. Hence they must be coded *before* the note at which the slur begins. The primary macro call is `\slur` p_1p_2sl where p_1 and p_2 are respectively the initial and final pitches, s is the sense, either `u` or `d`, and l is the length, in units of the current value of `noteskip`. Thus, thirds slured in pairs can be coded:

```
\NOtes\slur ced1\qu{ce}\en
\NOtes\slur dfd1\qu{df}\en
\NOtes\slur egd1\qu{eg}\en
```

```
\NOtes\slur{'e}cu1\ql{ec}\en
\NOtes\slur{'d}bu1\ql{db}\en
\NOtes\slur{'c}au1\ql{ca}\en
```

which yields:



Similarly, ties may be set using `\tie` *psl* where *p* is the (single) pitch, and the other parameters are as described above.

Variants on these macros are provided to change the slur length and vertical offset relative to the starting note, as follows.

- `\sslur` p_1p_2sl, which sets a 'short' slur, designed for the case when a slur is required to link notes which form part of a chord (see `\isslur` above). The macro `\stie` *psl* sets analagous ties.

2.12.5 Restrictions

Generation of some of the slur fonts is problematic for METAFONT. In particular, the long ties tend to exceed METAFONT's maximum dimension for high resolution printers. However, for normal 300 dpi printers there is no problem. A solution to this limitation is planned for Postscript© printers using DVIPS `\special` commands.

2.13 Bars

2.13.1 Bars and spacing

Ordinary *bars* are coded using the macro `\bar`. Its drawback is that it does not differ from `\bar` which is already defined in TEX's mathematical mode. Therefore, inside `\startpiece...\endpiece` *our* `\bar` means a musician *bar*[19] and outside, it keeps its original meaning. If you *really* need the original `\bar` inside, you can say `\endcatcodesmusic...\bar...\catcodesmusic`.

2.13.2 Bar numbering

Unless otherwise specified, bars are numbered. This is a good means of finding errors provided that the MusiXTEX user has put comments in his source text recording the (expected) bar number. However, this can look unpleasant for final outputs, since the habit is to number bars only each other five or ten bars. This is not a serious problem since the frequency of bar numbering is defined as:

```
\def\freqbarno{1}
```

If you replace the `1` by `5`, bar numbering will occur each other five bars. You can also inhibit any bar number[20] printing by telling:

```
\nobarnumbers
```

If you want to restore bar numbering after `\nobarnumbers`, you can say

[19] *Mesure* in French, *Takt* in German.
[20] default inside this manual

`\barnumbers`

The bar counter is also accessible, its name is `\barno`. You can change it without any dramatic consequence.

Sometimes there is need for to start with a `\barno` value which differs from 1. Perhaps the piece starts with an *upbeat*, then you can code

`\startbarno=0`

in front of `\startpiece`; or you may want to typeset only an excerpt starting for example with bar number 198, then you can code `\startbarno=198`. Pay attention because saying `\startbarno=`*n* has a global meaning. So if you want to `\input` several pieces whose starting bar numbers differ, you have to adjust the value of `\startbarno` each time.

The best compromise between information of the bar number and a pleasant lookout is the command `\systemnumbers` which draws the bar numbers only at the left top of every system (or staff? then the command will be renamed to `\staffnumbers` or `\staffbarnumbers`).

You can influence the height of the system numbers by coding

`\def\raisebarno{`*any TEX dimension*`}`

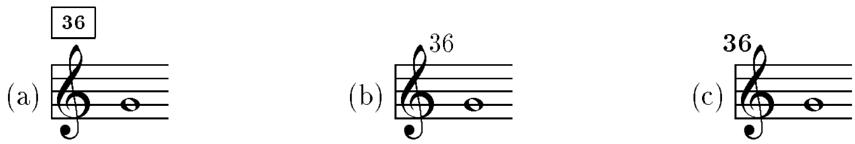
whose default is set to `4\internote` to fit above a violin clef. You can influence its horizontal position with

`\def\shiftbarno{`*any TEX dimension*`}`

which default value is `0pt`. If you don't like the box around the bar number, you can redefine the macro which sets the system bar number; its default is

`\def\writebarno{\boxit{\eightbf\the\barno\barnoadd}}`

where `\boxit` is a utility macro provided by MusiXTEX whose purpose is to enclose its argument in a $\boxed{\text{box}}$.



which was coded as

(a) (default)

(b)
```
\def\writebarno{\tenrm\the\barno\barnoadd}%
\def\raisebarno{2\internote}%
\def\shiftbarno{2.5\Interligne}%
```

(c)
```
\def\writebarno{\llap{\tenbf\the\barno\barnoadd}}%
\def\raisebarno{2\internote}%
\def\shiftbarno{1.3\Interligne}%
```

If the previous line does not stop with a bar rule to the next system bar number is added the `\writezbarno` whose default setting is the lower character `'a'`. If you want a `'+'` instead, you can say

`\def\writezbarno{+}`

Besides, you can suppress the messages of bar numbers on `stdout` (normally screen) with `\nobarmessage`. In the same way, you can suppress the messages abour new lines (new systems) with `\nolinemessages`.

2.13.3 Full and instrument divided bars

Normally, bars (as well as double bars, final bars and repeat bars) are drawn as a continuous line, starting for the bottom of the lower staff of the lower instrument, and ending at the top of the upper staff of the upper instrument. However, one may want to have discontinuous bars, that is, one continuous bar for all the staffs of a unique instrument. This is done by issuing the command `\sepbarrules`. An example of this is given in the `ANGESCAO` (or `ANGESCAM`) example; it has also been used in the example of section 2.19.2.

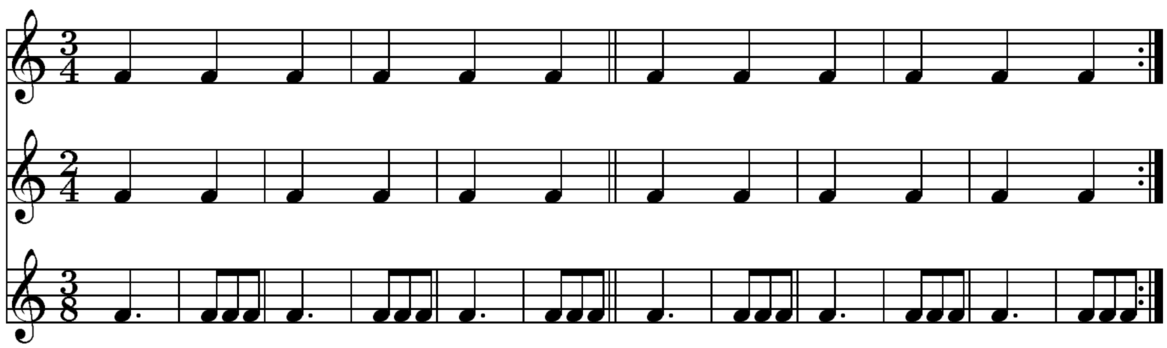
The initial situation can be forced or restored by `\stdbarrules`. In the extension library are some more types of bar rules, mainly for very old music, see 2.26.12.

2.13.4 Instruments with no coherent bar division

In some special scores with several instruments, it may happen that distinc instruments, not only have different meters (e.g. 2/4 and 6/8 to avoid printing triolet codes for some instrument), but also have bar lines not synchronized. Obviously, this can be implemented ***only if*** `\sepbarrules` has been stated. Then, five specific commands can be issued :

- `\hidebarrule` n hides the bar rule for instrument n, until this is changed by `\showbarrule` n.
- `\showbarrule` n stops hiding the bar rule for instrument n, until this is changed by `\hidebarrule` n.
- `\Hidebarrule` n hides the bar rule for instrument n, only for the next bar.
- `\Showbarrule` n exceptionally shows the bar rule for instrument n, and then resets intrument n to `Hidebarrule`.
- `\showallbarrules` resets all defined instruments to `showbarrule` n. This command is automatically inserted with double bars, final bars and repeats.

Thus, the following example :



was obtained with the following coding

```
\instrumentnumber3
\setmeter3{{\meterfrac{3}{4}}}
\setmeter2{{\meterfrac{2}{4}}}
\setmeter1{{\meterfrac{3}{8}}}
\nobarnumbers
\sepbarrules
```

```
\startextract
\NOtes\pt f\qa f&\qa f&\qa f\en
\hidebarrule2\hidebarrule3\bar
\NOtes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\showbarrule2\bar
\NOtes\pt f\qa f&\qa f&\qa f\en
\hidebarrule2\showbarrule3\bar
\NOtes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\showbarrule2\hidebarrule3\bar
\NOtes\pt f\qa f&\qa f&\qa f\en
\hidebarrule2\bar
\NOtes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\setdoublebar
\bar\hidebarrule3
\NOtes\pt f\qa f&\qa f&\qa f\en
\Hidebarrule2\bar
\NOtes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\bar
\NOtes\pt f\qa f&\qa f&\qa f\en
\message{Showbarrule3 coming}%
\Hidebarrule2\Showbarrule3\bar
\NOtes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\bar
\NOtes\pt f\qa f&\qa f&\qa f\en
\Hidebarrule2\bar
\NOtes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\setrightrepeat
\endextract
```

2.14 Managing the layout of your score

2.14.1 Line and page breaking

Those who usually worked with MusicTEX before must be aware that this has been deeply changed in MusiXTEX. All line-breaking decisions[21] are done by the external program `musixflx`. Then you get the most even results when you insert the fewest *manual* line breaks possible. But sometimes there is really need for them, e.g. page break should preferably occur when the musician has one hand free to turn the sheet.

You can force a line break with `\alaligne` instead of `\bar`. In the same way, you can code `\alapage` to force an `\eject` with proper reinitialization of staffs, clefs and signatures.

On the other hand, you may want to forbid line-breaking at a bar, then you should replace `\bar` with `\xbar`.

Conversely, you may want to break a line *not at a bar*[22]. This is allowed by `\zbar` (optional line break) or forced by `\zalaligne` or `\zalapage`.

[21] With exception of the MusicTEX command `\autolines`, which is provided in the file `musixcpt.tex`, see page 82.

[22] For example, you may prefer to turn the page at a place where the pianist has one hand free, in the middle of a bar.

The heavy final double bar of a piece is provided by `\Endpiece` or `\Stoppiece`. If you just want to terminate the text with a simple bar, you say `\stoppiece` or `\endpiece`. If you want to terminate it without a bar, you code `\zstoppiece`.

Once you have stopped the score by any of these means, you can restart it using `\contpiece`. If you want to restart indenting the next line (system) by the dimension `\parindent`, then you should rather use `\Contpiece`, which is recommended when changing the the number of instruments within a single piece.

However, neither `\contpiece` nor `\Contpiece` can be used if you change the signature (i.e. using `\generalsignature`) between `stoppiece` and `\contpiece`), since the space used by signatures is not constant and it has to be taken in account, e.g. by some `\changecontext`. In that case, you have to restart your score using `\startpiece`, which in turn requires you to save `\barno` to its furtner starting value `\startbarno` if you do not want it reset at 1, and to manage the instrument names and/or the `\parindent` if necessary.

If you want the next vertical bar to be a double bar, you have to declare `\doublebar` or `\setdoublebar` before the `\bar` (or the `\stoppiece` or `\alaligne` or `\alapage`) to be marked with a double thin bar. In the same way you can declare `\setdoubleBAR` if you want to have a heavy double bar (the same as `\Stoppiece`) and even `\setemptybar` to make the next `\bar` invisible[23].

2.14.2 How to manage individual page layout

1. You write your own output routine, which centers the contents in the middle of the page. Most important are then the values of `\stafftopmarg` and `\staffbotmarg`, because they decide of the amount of margin between successive systems.

2. If you say `\raggedbottom`, the vertical glue is removed and the score is rather compacted at the top of page.

3. You change the value of `\parskip` and say `\normalbottom` (which behaves like LaTeX's `\flushbottom` and which is done by default working with `plain.tex`). Then the vertical space between the staffs is changed to get the first staff on page on top and the last staff on page to bottom (depends on `\vsize`). It might be clever to insert a `\eject` before the `\bye`. `\musicparskip` set between two systems no extra distance, but the possibility to increase the distance up to `5\Interligne`.

Besides, following values of e.g. paper size are changed (only if you *do not* work with LaTeX) to:

```
\parindent= 0pt
\hoffset= -15.4mm
\voffset= -10mm
\hsize= 190mm
\vsize= 260mm
```

2.14.3 How to adjust the global line and page layout

Once you have made your whole score, you will probably find out that all systems (all "lines") have a correct layout with nice beams, nice slurs, and others. But it is likely that:

- The result takes too many or too few pages and you estimate a more or less compact score would be suitable.

[23]These latter features are given for your information, but they should be used only in case of emergency.

- The results takes a convenient number of pages, but the last page exhibits a widow line or an ugly blank space at the end.

To solve this, you may revert to two strategies:

1. Explicitely force line and page breaking at the very places you like; this can be done using `\alaligne`, `\alapage`, `\zalaligne` or `\zalapage` commands. This can also be done using the `\autolines` command imported from MusicTEX and provided in the `musixcpt.tex` additional file. But this way of doing may result in unfortunately uneven distribution of the notes in the score, which is therefore not very smart.

2. Adapt both the `\mulooseness` and the global value of `\elemskip`: increasing `\mulooseness` (whose default is 0) increases the total number of systems, and if you are clever you may be able to fill an integer number of pages. Besides, increasing the stated `\elemskip` (use `\showthe\elemskip` to find its default value) also makes note spacing wider, and its advantage is that you can tune it by half a percent if needed.

   REMARK: `\elemskip` *can be retrieved in both the first and the third pass (second TEXing pass), but any assignment of a value (a dimension value) to* `\elemskip` *will be of no effect at third pass, since it is taken from the* `*.mx2` *file yielded by* `musixflx`*. On the contrary, assigning a value to* `\noteskip` *within* `\notes...\enotes` *is efficient in both passes, and cause some weird errors if you mix up scalable and not scalable dimensions.*

2.15 Changing score attributes

As seen before, you can change the signature of the whole set of instruments by `\generalsignature` n where $n > 0$ means a number of sharps, $n < 0$ means a number of flats. Or, you may prefer to change the signature of only one or two instruments by the statement:

`\setsign` n{s}

where n is the number of the instrument considered, and s its specific signature. Since you may change simultaneously (with respect to the score) but consecutively (with respect to your code) the signatures of several instruments, this change takes place only when you say `\changesignature` (within a bar) or `\changecontext` (after a single vertical rule) or `\Changecontext` (after a double vertical rule) or `\zchangecontext` (without a bar rule[24]).

Normally, changing a signature from flats to sharps or sharps to flats or reducing the number of sharps/flats will produce the convenient set of naturals to emphasise what is suppressed. This standard feature can be temporarily inhibited by the command `\ignorenats` to be issued before the next `\changecontext` or `\changesignature`.

In the same way, you may want to change the active clefs. This is done by

`\setclef`{n}{$s_1 s_2 s_3 s_4$}%

where r is the number of the instrument, s_1 specifies the clef of the lower staff, s_2 the clef of the second staff, etc. $s_1 = 6$ means the *bass* clef (clef de fa in French), $s_1 = 0$ means the *violin* clef (clef de sol in French), $s_1 = 1$ through $s_1 = 4$ mean the *alto* clef (clef d'ut in French) set on first (lower) through fourth (next to upper line of the staff), $s_1 = 5$ or $s_1 = 6$ mean the bass clef on the third or fourth line respectively.

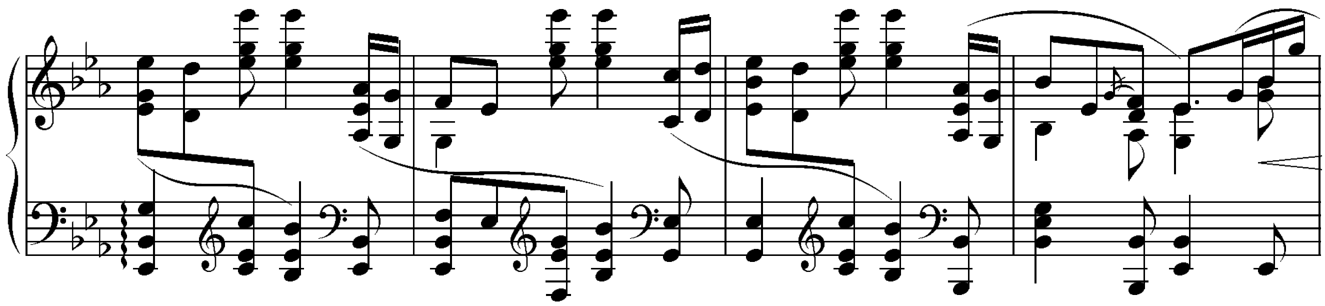
As seen above in the case of signatures, several clefs may be changed at the same time; thus all the clef changes become operational only when the macro `\changeclefs` is coded. Normal

[24]Old composers uses the bar rules very rarely

usage consists in issuing this command before the bar, not after (this helps the music player when the change happens across a line break).

The `\changeclefs` command normally takes some horizontal space to put one of more clef symbols, but it may happen that you have no notes immediately before on the staff whose clef is changed. In that case, you can use `\zchangeclefs` which performs the same posting, overwritten left on the last part of the score in that staff. Of course it is your responsibility to ensure that no notes will collide with the clef change symbols.

The MusicTEX problem if a clef change is coded while one or several beams are pending has been removed in MusiXTEX. As an enhanced example, we provide an excerpt of BRAHMS's Intermezzo op. 117,1 initially coded by Miguel FILGUEIRAS:



The first bar of this excerpt was coded as:

```
\begin{music}
\instrumentnumber{1}
\setstaffs1{2}
\interstaff{11}
\setclef1\bass % F- and G-clefs
\generalsignature{-3}% 3 flats
\parindent0pt
\startpiece
\qspace\nspace
\Notes\arpeggio E5\ibslurd0o\zq{EI}\qu N%
  \nextstaff\ibl0e{-2}\zq{eg}\qb0l\zq d\qb0k\enotes
\setclef1\treble\zchangeclefs
\nspace
\Notes\tbu0\zq{ce}\qb0j\nextstaff\zq{sn}\cl l\enotes
\notes\invertslur0\tslur0t\zq{be}\qu i\nextstaff\zq{sn}\ql l\enotes
\setclef1\bass \changeclefs
\notes\islurd0k\zq E\cu I\nextstaff
  \ibbu1h{-3}\zq{ae}\qb1h\tbu1\zq N\qb1g\enotes
\end{music}
```

Meter changes are implemented in following way:

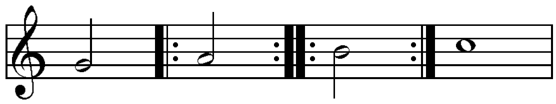
`\setmeter` n`{{`$m1$`}{`$m2$`}{`$m3$`}{`$m4$`}}%`

where n is the number of the instrument, $m1$ specifies the meter of the lower staff, $m2$ the meter of the second staff, etc. (if necessary!).

Since meter changes are meaningful only across bars, they are actually taken in account with `\changecontext` or `\Changecontext` or `\zchangecontext` or `\alaligne` or `\alapage`.

2.16 Repeats

To insert a *repeat bar* you can use following sets of procedures, namely `\leftrepeat`, `\rightrepeat` and `\leftrightrepeat`, which are substitute the `\bar`. For example:



has been coded as:

```
\NOTes\ha g\enotes
\leftrepeat
\NOTes\ha h\enotes
\leftrightrepeat
\NOTes\ha i\enotes
\rightrepeat
\NOTEs\wh j\enotes
```

Special cases are forced line breaks: if you want to force a new line at a repeat, you should code respectively:

```
\setrightrepeat\alaligne
\setrightrepeat\endpiece
\alaligne\leftrepeat
\contpiece\leftrepeat
\startpiece\leftrepeat
```

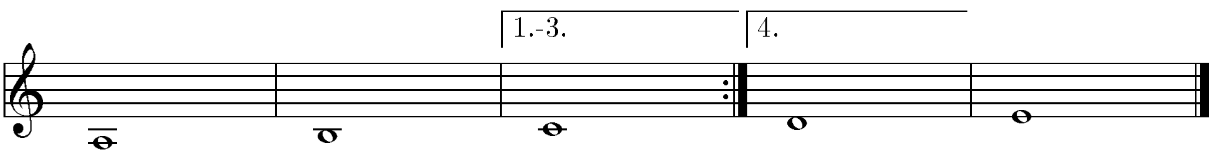
or the combination of two of these in the case of a left/right repeat.

A second way of coding consists in saying `\setleftrepeat`, `\setrightrepeat` or `\setleftrightrepeat` before a bar (`\bar`), `\stoppiece` or `\changecontext`). In this case, the next single vertical bar will be replaced with the selected repeat bar. This meets the traditional music typesetting conventions in the only case of the *right repeat* but, unfortunately, left and left/right repeats use to behave in a different manner when in the middle of a line and at a line break.

2.16.1 Specific first and second pass scoring

A frequent situation consists in a long part of score repeated two or several times, but its last few bars are different at first pass and at second pass. This can be specified by saying `\setvolta{`*text*`}` before the `\bar` beginning the part specific to that first or second part[25].

Then, only one bar has a horizontal rule above. No termination command is needed since termination is implied by the following `\bar` or repeat symbol. For example:

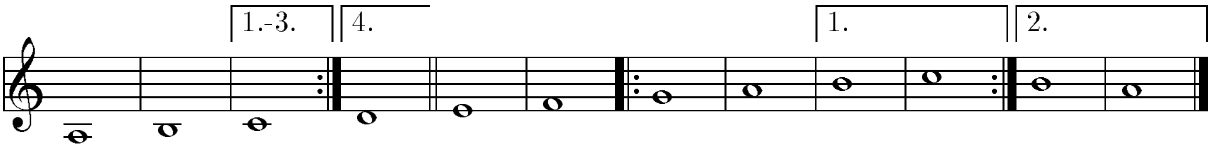


which was coded as:

[25] The word "volta" comes from Italian, it means "time" in the sense of "first time", "second time" or "pass" in that case.

```
\parindent0pt \startpiece \addspace\afterruleskip
\NOTEs\wh a\en\bar
\NOTEs\wh b\en\setvolta{1.-3}\bar \NOTEs\wh c\en\setvolta4\rightrepeat
\NOTEs\wh d\en\bar
\NOTEs\wh e\en\Endpiece
```

If the music typesetter wants to include more than one bar in the first pass[es] section and he wants all these to have a continuous line above, then he can code `\Setvolta{`*text*`}` before the bar he wants the "volta" to begin, and put `\setendvolta` or `\setendvoltabox` before the bar they should end. `\setendvoltabox` makes the upper line to be terminated with a descending hook, symmetrical to the opening small vertical bar.



has been coded as:

```
\startpiece \addspace\afterruleskip
\NOTEs\wh a\en\bar
\NOTEs\wh b\en\setvoltabox{1.-3}\bar
\NOTEs\wh c\en\setvolta4\setendvolta\rightrepeat
\NOTEs\wh d\en\doublebar
\NOTEs\wh e\en\bar
\NOTEs\wh f\en\leftrepeat
\NOTEs\wh g\en\bar
\NOTEs\wh h\en\Setvolta1\bar
\NOTEs\wh i\en\bar
\NOTEs\wh j\en\Setvolta2\setendvoltabox\rightrepeat
\NOTEs\wh i\en\bar
\NOTEs\wh h\en\setendvoltabox
\Endpiece
```

The "volta" symbols are normally set at `4\internote` above the upper line of the staff and one bar(length?) wide.

The final down hook is stated, either by `\setendvoltabox` at the end, or `\setvoltabox` at the beginning.

IMPORTANT: You are no more allowed to open `\Setvolta` *or* `\setvoltabox` *whan another of these multi-bar volta is pending. To re-open a* `\Setvolta` *at the ens of another one, you must explicitly order the first one to be closed at the end of the bar where the second one is opened. Given examples had to be corrected in order to meet this new safety requirement.*

For the sake of portability, `\endvolta` is identical ti `\setendvolta`, and `\endvoltabox` is identical to `\setendvoltabox`. Theses changes in the names ar intended to recall that these commant take effect only at the next bar, repeat or line break.

The altitude of the "volta" above the upper staff can be changed, redefining `\raisevolta`, e.g.:

```
\def\raisevolta{7\internote}
```

or define it to be any other valid dimension.

REMARK: the dot after "volta" numbering can be removed, simply by saying:

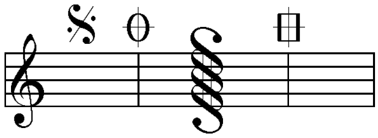
```
\def\voltadot {}
```

and reset by:

```
\def\voltadot {.}
```

2.16.2 Large scope repeats and orientation marks

Large scope repeats have also been provided with special symbols, namely `\coda` *p*, `\Coda` *p*, `\segno` *p*, where *p* specifies the pitch. A bigger symbol is `\Segno` with no argument. For example, the following figure



has been coded:

```
\NOtes\segno m\enotes\bar
\NOtes\coda m\enotes
\NOtes\Segno\enotes\bar
\NOtes\Coda m\enotes
```

Orientation marks are set more often above larger orchestral pieces for easy come together of all instruments for exercise reasons. Often used are circled or boxed uppercase characters or digits. This can be done with

`\boxit{`*text*`}` or
`\circleit{`*text*`}`

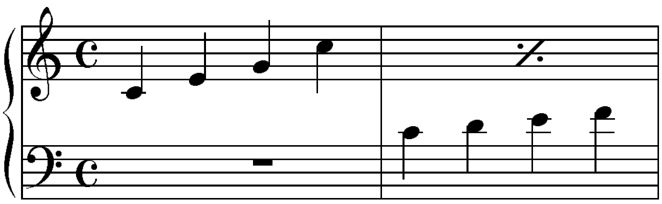
The distance between the box and the including text is influenced by

`\boxitsep=`*any TEX dimension*

which is setup by default to `3pt`. The usage is recommended with `\Uptext`, `\zcharnote`, `\zchar` or `\ccharnote`.

2.16.3 Repeating the last bar

This can be done using the symbol `\duevolte` (often used with `\centerbar`, see example in 2.7.3, p. 32), e.g.:



whose coding is:

```
\generalmeter\meterC
\setclef1\bass\setstaffs1{2}
\startextract
\NOtes\sk\sk\pause|\qa{cegj}\en
\bar\NOtes\qa{cdef}|\sk\sk\duevolte\en
\endextract
```

2.17 Miscellaneous

2.17.1 Putting anything anywhere

Special macros are provided to help the composer to set any TEX text on the staffs. The macro

`\zcharnote` *p*{text}

sets the given text with its base line at pitch *p* of the current staff (this means it must be coded inside `\notes...\enotes`). Whatever the length of the text, no spacing occurs. If you want the possible spilling text to expand on the left rather than on the right, then you can use `\lcharnote`. If you want the possible spilling text centered to current position, then you can use `\ccharnote` which causes no space.

The macro `\zcharnote` is fit for coding special notations like accents above or below the notes. Also available are `\zchar`, `\lchar` and `\cchar`, which allows only numbers for pitch, but enables the use of floats too.

To place some text at the mid-position between the two staffs of a keyboard instrument, you may code:

`\zmidstaff{`*text*`}%` (to right)
`\lmidstaff{`*text*`}%` (to left)
`\cmidstaff{`*text*`}%` (centred)

being however careful, a) to put it inside `\notes...\enotes`, b) to code it in the text of the lower staff.

A text to be put above the current staff is introduced by `\uptext{...}`. This may however cause some collision with bar numbering or notes above the staff; it is then wise to use `\Uptext{...}` which puts the text two note line distances higher (recommended to post the tempo).

2.17.2 Fonts

The text fonts loaded by MusiXTEX come in six different type sizes and three styles. The type sizes are 8 pt, 10 pt, 12 pt, 14 pt, 17 pt and 25 pt, while the three styles are roman, bold and italic. The three smaller type sizes are available in all three styles, while the larger three sizes, which are intended for titles, are available only in bold style. The size selection macros are respectively `\smalltype`, `\normtype`, `\medtype`, `\bigtype`, `\Bigtype` , `\BIgtype` and `\BIGtype`. Following the size selection, the style may be selected or changed using `\rm` (roman), `\bf` (bold) or `\it` (italic). If no style selection is made, roman style will result for the sizes smaller than `\medtype`. For the larger sizes, style selection is not required since only bold style is provided. Thus, selection of eight point italic is done using `\smalltype\it`, while twelve point roman is selected using `\medtype\rm` or simply `\medtype`. To change between styles while maintaining the same size, code `\rm`, `\it` or `\bf` as in Plain TEX. This information is summarized in the following table.

Size, pt	Size selection	Style selection
8	`\smalltype`	`\rm, \bf, \it`
10	`\normtype`	`\rm, \bf, \it`
12	`\medtype`	`\rm, \bf, \it`
14	`\bigtype`	`(\bf)`
17	`\Bigtype`	`(\bf)`
20	`\BIgtype`	`(\bf)`
25	`\BIGtype`	`(\bf)`

Two other text fonts are provided for dynamic markings. These are `\ppffsixteen`, `\ppfftwenty` and `\ppfftwentyfour`, suitable for dynamic markings with staff sizes of 16 pt, 20 pt and 24 pt respectively. The appropriate font for the current staff size may be selected by coding `\ppff`.

Naturally other fonts may be loaded by the user if required. When MusiXTEX is started, the default text font is ten point roman, equivalent to `\normtype\rm`.

2.17.2.1 Examples

Size and style	Example
`\smalltype`	small roman
`\smalltype\bf`	**small bold**
`\smalltype\it`	*small italic*
`\normtype`	normal roman
`\normtype\bf`	**normal bold**
`\normtype\it`	*normal italic*
`\medtype`	medium roman
`\medtype\bf`	**medium bold**
`\medtype\it`	*medium italic*
`\bigtype`	**big bold**
`\Bigtype`	**Big bold**
`\BIgtype`	**BIg bold**
`\BIGtype`	**BIG bold**
`\ppffsixteen`	***pp ff diminuendo***
`\ppfftwenty`	***pp ff crescendo***
`\ppfftwentyfour`	***pp ff crescendo***

2.17.3 Metronomic indications

Metronomic indication deserves a special macro. The mention:

𝅗𝅥. = 60

is coded by `\metron{\hup}{60}` (normally embedded in `\Uptext` which is in turn embedded within `\notes...\enotes`).

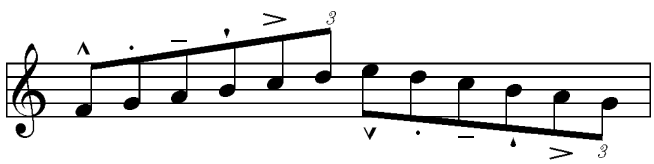
2.17.4 Accents

- `\upz` p (upper *pizzicato*) to put a dot above a note head at pitch p,
- `\lpz` p (lower *pizzicato*) to put a dot below a note head at pitch p,
- `\usf` p (upper *sforzando*) to put a $>$ accent above a note head at pitch p,
- `\lsf` p (lower *sforzando*) to put a $>$ accent below a note head at pitch p,
- `\ust` p (upper *staccato* or *portato*) to put a hyphen above a note head at pitch p,
- `\lst` p (lower *staccato* or *portato*) to put a hyphen below a note head at pitch p,
- `\uppz` p (upper strong *pizzicato*) to put an apostrophe above a note head at pitch p,

- `\lppz` p (lower strong *pizzicato*) to put a reversed apostrophe below a note head at pitch p.
- `\usfz` p (upper *sforzato*) to put a 'roof' above a note head at pitch p,
- `\lsfz` p (lower *sforzato*) to put a reversed 'roof' below a note head at pitch p.
- `\upzst` p (upper *portato/staccato*) to put a combined portato/staccato sign above a note head at pitch p,
- `\lpzst` p (lower *portato/staccato*) to put a combined portato/staccato sign below a note head at pitch p.
- `\flageolett` p to put a thin circle above a note head at pitch p.
- `\upbow` to indicate a bowing for strings in upper direction.
- `\downbow` opposite to `\upbow`.

Because whole notes (breve, arbitrary, ...) have a different note head width the accents appear not centered above them. Therefore you can use `\wholeshift{`*text*`}` which centers accents and others which are centered above a quarter note head to appear centered above a whole note. This is used for e.g. `\Fermataup`.

Also available are the variants[26] of the most used accents which will be automatically positioned above or below a beam. Therefore the known accents are preceeded with the letter 'b' and, instead of the pitch, the beam reference number is appended as an argument. Thus



was coded as:

```
\startextract
\Notes\ibu0f3\busfz0\qb0f\bupz0\qb0g\bust0\qb0h%
  \buppz0\qb0i\busf0\qb0j\butext0\tqh0k\en
\Notes\Ibl0lg5\blsfz0\qb0l\blpz0\qb0k\blst0\qb0j%
  \blppz0\qb0i\blsf0\qb0h\bltext0\tqb0g\en
\endextract
```

2.17.5 *Indication of x-tuplets

Triplets and other x-tuplets — in practice the italic numbers *3*, *4*, *5*, etc. can be set at the right horizontal/vertical position invoking the `\triolet` or `\xtuplet` macros at the leftmost position of the group, i.e. before the first note is coded.

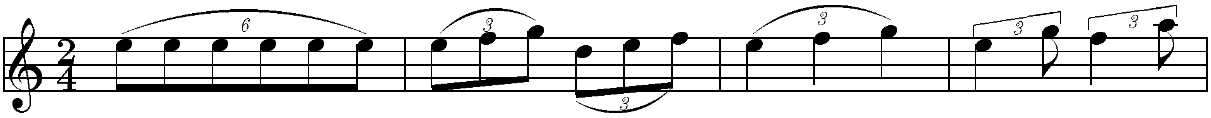
`\triolet` has only one argument, the pitch of the bottom of the figure 3, `\xtuplet` has two arguments: the figure and its pitch.

If the figure has to be put above an upper beam or below a lower beam, using the macros `\butext` (above) and `\bltext` (below) is advisable to post the figure above or below the given beam. `\butext` and `\bltext` have one argument, namely the beam number (namely the same

[26]Thanks to Klaus BECHERT's corrections.

as the first argument of commands such as `\ibu` or `\Ibu`). If the group of notes is not a triolet, the actual figure has to be changed, redefining the macro `\txt` which is set up by default to:

```
\def\txt{\eightit 3}%
```




```
\notesp\xtuplet6n\isluru0l\ibl0l0\qb0{lllll}\tslur0l\tqb0l\en\bar
\notesp\triolet n\isluru0l\Ibl0ln2\qb0{lm}\tslur0n\tqb0n\en
\notesp\ibslurd0k\Ibl0km2\qb0k\bltext0\qb0l\tdbslur0m\tqb0m\en\bar
\Notesp\triolet o\isluru0l\ql{lm}\tslur0n\ql n\en\bar
\Notesp\uptrio o16\ql l\en\notesp\cl n\en
\Notesp\uptrio p16\ql m\en\notesp\cl o\en
```

2.17.6 *Usual ornaments

2.17.6.1 Arpeggios

Arpeggios (i.e.) can be coded with the macro

`\arpeggio` *pm*

where *p* is the pitch of the base of the arpeggio symbol and *m* is its multiplicity (one period is equal to one space between staff lines, i.e. 5 points). This macro causes no spacing. If should be issued before the concerned chords. Its variant `\larpeggio` sets the arpeggio symbol nearly one note head width on the left, in order to avoid collision with accidentals in front of the chords.

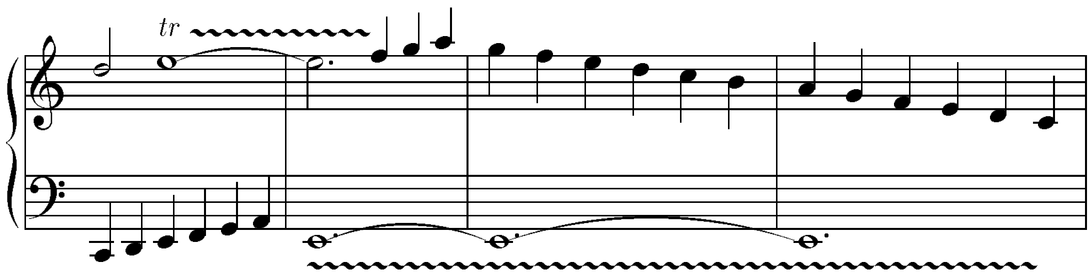
2.17.6.2 Trills

Trills can be coded in several ways. `\trille` *pl* (where *p* is the pitch and *l* is a number of current *noteskip*) yields while `\Trille` *pl* yields *tr* .

For longer trills expanding over bars and line breaks, a better way consists in specifying the beginning with `\Itrille` *np* where *n* is the trill reference number ($0 \leq n < 6 =$`\maxtrills`) and terminate it with the command `\Ttrille` *n*. If one wants the *tr* mention at the beginning, then one should use `\ITrille` *np*. The ancient commands `\itrille`, `\ttrille` *p* and `\tTrille` *p* still work but beware that their syntax is different, that the opening and the closing must be in the same bar and that the use the same registers as `\Itrille 0`.

REMARK: `trille` *and* `Trille` *also exist in MusicTEX. Unfortunately, the same macro names do not have the same syntax and the same semantic in both packages and* `musixcpt.tex` *forces the behaviour and the syntax of MusicTEX. A solution to this problem consists in using* `\trilleC` *and* `\TrilleC` *to have the MusicTEX behaviour, and* `\trilleX` *and* `\TrilleX` *to have the MusiXTEX behaviour, even when* `musixcpt` *is invoked.*

For example:

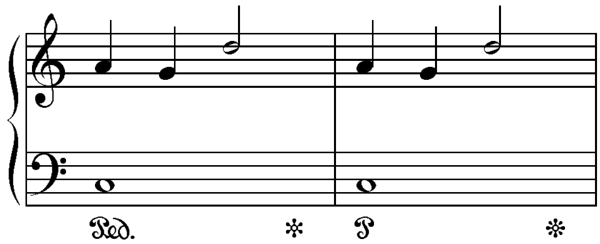


whose coding is

```
\begin{music}
\instrumentnumber{1}
\setstaffs12
\setclef1{6000}
%
\startextract
\notes\qu{CDEFGH}|\hu k\sk\ITrille 1p\itenu1l\wh l\enotes
\bar
\notes\Itrille 2A\itenu1e\whp E|\tten1\hlp l\sk\Ttrille 1\qu {mno}\enotes
\bar
\Notes\tten1\itenu1E\whp E|\ql{nmlkji}\Toctfin1\enotes
\bar
\Notes\tten1\whp E\sk\sk\sk\sk\Ttrille2|\qu{hgfedc}\enotes
\endextract
```

2.17.6.3 Piano pedal commands

`\PED` inserts the piano pedal command below the staff; pedal release is specified by `\DEP`; also available are shorter variants of the pedal signs which are invoked with `\sPED` and `\sDEP`; thus the following example



was coded as:

```
 \NOtes\PED\wh J|\qu
h\enotes
 \NOtes|\qu g\enotes
 \NOtes|\hu k\enotes
 \Notes\DEP\enotes \bar
 \NOtes\sPED\wh J|\qu h\enotes
 \NOtes|\qu g\enotes
 \NOtes|\hu k\enotes
 \Notes\sDEP\enotes
```

The vertical position of `\PED`, `\sPED`, `\DEP` and `\sDEP` can be globally changed by redefining its elevation, which is setup by default to

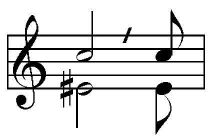
```
\def\raiseped{-5}%
```

If you only want to change a few of them, you can use the more fundamental macros `\Ped`, `\sPed`, `\Dep` and `\sDep` in combination with `\zchar` or `\zcharnote`.

Since the 𝔓𝔢𝔡. symbol is rather wide, it might collide with bass notes in an ugly way. Then a solution consists in shifting if to the left, by coding `\loff{\PED}`.

2.17.6.4 Other ornaments

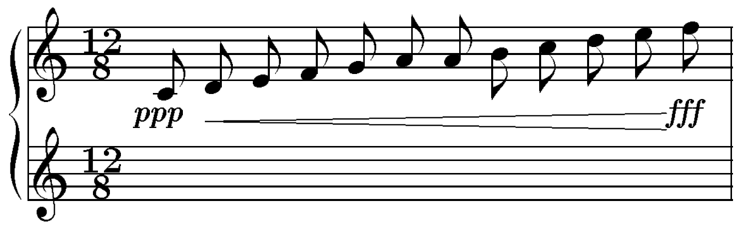
- `\mordent` *p* for ,
- `\Mordent` *p* for ,
- `\shake` *p* for ,
- `\Shake` *p* for ,
- `\Shakel` *p* for ,
- `\Shakesw` *p* for ,
- `\Shakene` *p* for ,
- `\Shakenw` *p* for ,
- `\turn` *p* for ,
- `\backturn` *p* for ,
- `\fermataup` *p* puts a *fermata* at pitch *p*. No spacing occurs.
- `\fermatadown` *p* puts a reverse *fermata* at the same place.
- `\Fermataup` *p* puts a *fermata* at pitch *p* centered above a whole note. No spacing occurs.
- `\Fermatadown` *p* puts a reverse *fermata* at the same place.
- A big *breathing* comma can be put above the staff — to indicate where the singer is welcome to breath, or to indicate a short rest — using `\zbreath` (no spacing) or `\cbreath` (centred in a `\noteskip` space). The MusicTEX `\breath` is no more available since it involved a “hard” spacing of the width of that character, which resulted in troubles with MusiXTEX.
- The `\caesura` command can also be used to insert a small slash, such as



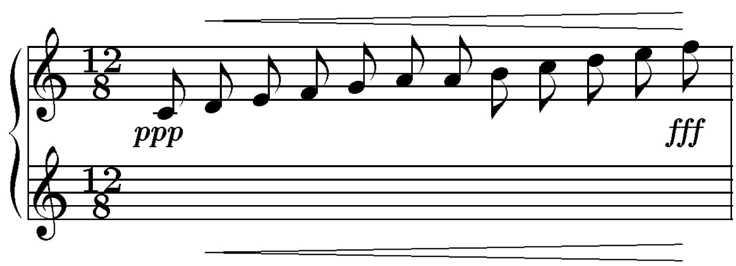
2.17.7 Dynamic signs

Various dynamic signs can be posted with `\crescendo{`*l*`}` or `\decrescendo{`*l*`}`, to be used as arguments to `\zcharnote`, `\zchar`, `\uptext`, `\zmidstaff`, etc.

Note: The biggest sign is $\simeq$ 68 mm long.

Alternately they can be produced using the pair of `\icresc` and `\tcresc` or `\tdecresc`. Saying one time `\icresc` you can force several `\tcresc` or `\tdecresc` with the same starting abscissa. Tha altitude of the [de]crescendo symbol is specified, not by the `\icresc` macro which only defines the stating abscissa, but by the `\tcresc` or `\tdecresc` which is normally lifted up or down by means of a `\zmidstaff` or a `\zcharnote` command whose first argument has to be adjusted according to your visual needs.

which was coded as:

```
\Notes\cmidstaff\ppp|\ca c\en
\Notes\icresc|\ca{defgh'abcde}\en
\Notes\zmidstaff{\loff\tcresc}\cmidstaff\fff|\ca{'f}\en
```



which was coded as:

```
\Notes\cmidstaff\ppp|\ca c\en
\Notes\icresc|\ca{defgh'abcde}\en
\Notes\zcharnote N{\tcresc}\cmidstaff\fff|\zcharnote q{\tcresc}\ca{'f}\en
```

Intensity conventional signs are: `\ppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\fp`, `\sF`, `\ff`, `\fff`, `\ffff`, resulting in ***pppp***, ***ppp***, ***pp***, ***p***, ***mp***, ***mf***, ***f***, ***fp***, *sf*, ***ff***, ***fff*** and ***ffff***.

2.17.8 Length of note stems

Normally, the length of note stems is the distance of one octave, i.e. `7\internote`. This is equivalent to 4.66 `\interbeam`. The length of the stems may be changed using the macro `\stemlength`, e.g.:

```
\stemlength{5.2}
```

which will set the stemlength to 5.2 `\interbeam`. The default is `\stemlength{4.66}`.

Choral music with four voices is often typeset into two staffs. Thus, each staff holds two voices, one being typeset with stems up, the other with stems down. If the command `\stemcut` has been previously issued, stems that are outside the staff are cut depending on the pitch of the notes. If this is not wanted, this feature may be suppressed with the command `\nostemcut`. The default behaviour is `\stemcut` in plain MusiXTEX, and `\nostemcut` when `musixcpt.tex` is invoked.

Normally, down stems never end above the middle line of the staff and up stems never below that line. This extension of the stems may be inhibited for the next generated stem using the command `\stdstemfalse`. There is no `\stdstemtrue` since `\stdstemfalse` is reset after generating each new stem.

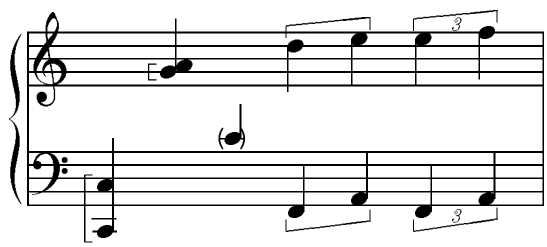
2.17.9 *Brackets and parentheses

Several brackets and parentheses have been provided for musical typesetting needs. These are:

- `\lpar{`p`}` and `\rpar{`p`}` which yield small parenthesis to enclose notes. These are used for *cautionary* accidentals but more comprehensive macros fulfill that specific purpose (see 2.10)
- `\bracket{`p`}{`n`}` to post square brackets of width n-internotes at left of a chord. Usually these brackets are used on keyboard instruments to specify, either two neighbouring notes played with a single finger, or some chord notes to be played with the alternate hand.
- `\doublethumb{`p`}`, to indicate a bracket of height 2`\internote`.
- `\ovbkt{`p`}{`n`}{`s`}` and `\unbkt{`p`}{`n`}{`s`}` draw a bracket over the music starting at the current position at pitch p, width $n\times$`noteskip` and slope s ($1 \sim 1.125$[degree]). causing no space.
- `\uptrio{`p`}{`n`}{`s`}` and `\downtrio{`p`}{`n`}{`s`}`, which are like `\ovbkt` but with free definable `\txt` centers in the middle.
- `\varline` h ℓ s builds an oblique line of height h (a dimension), of length ℓ and of slope s. Used to build oblique brackets.

  This feature is particularly useful for people who want to typeset *baroque* music using the ancient *ornament codings* rather than the modern equivalents. As a compromise, some macros kindly provided by Ian Collier in Great Britain have slightly been updated and provided.

For example:




```
\begin{music}
\setstaffs1{2}
\setclef1{\bass}
\startextract
\NOtes\bracket C8\zq C\qu J\en
\NOtes|\doublethumb g\rq h\qu g\en
\NOtes\lpar c\rpar c\qu c\en
\NOtes\unbkt C15\qu {FH}|\ovbkt n14\ql{kl}\en
\NOtes\downtrio C16\qu {FH}|\uptrio o14\ql{lm}\en
\endextract
\end{music}
```

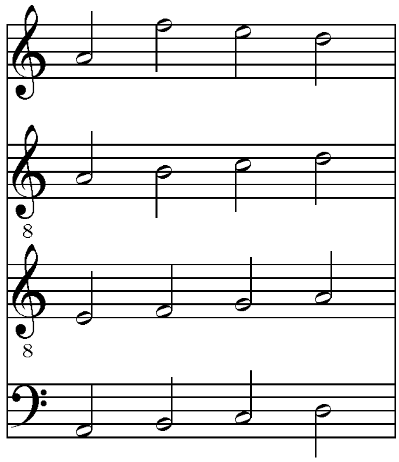
REMARK: if you used `\input musicext` *before, pay attention to the commands* `\ovbkt` *and* `\unbkt`, *their parameter meaning has changed from MusicTEX to MusiXTEX.*

2.17.10 New line synchronization of coding

The procedure named `\everystaff` is executed each time a new system is typed. It is normally void, but it can be defined (simply by `\def\everystaff{...}`) to tell MusiXTEX to post anything reasonable at the beginning of each system. It was used in the example NWIDOR to post octaviation dashed lines at the end of the piece.

Caution should be exercised to call `\everystaff` before `\startpiece` if its effect is needed for the first system of the score.

The procedure named `\atnextline`, normally void, is executed at the next computed or forced line break (using `\alaligne` or `\alapage`). More precisely, it is executed after the break and before the next system is typed. Thus it is fit for posting new definitions of layout parameters, when no system is pending[27].

In some scores, tenor parts are not coded using the *bass* clef, but using rather the *violin clef* subscripted by a `8`. This is not directly supported by the `\setclef` command, but it can be handled using `\everystaff` and `\zcharnote`. As an example the following score

was coded as:

```
\instrumentnumber{4}
\setclef1\bass
\def\everystaff{%
  \znotes&\zchar{-6}{\eightrm \kern -2\Interligne 8}%
  &\zchar{-6}{\eightrm \kern -2\Interligne 8}\en}%
\startextract
\NOTes\ha{HIJK}&\ha{efgh}&\ha{hijk}&\ha{hmlk}\en
\endextract
```

Caution: *the* `\everystaff` *must be called before* `startpiece` *in order to have that subscript at the first staff.*

2.18 Small and tiny notes

Before entering details, let us point out that we are presently concerned with typing notes of smaller size than the normal one, without attempting to change the interval between the five lines building a single staff. Changing staff line interval will be treated in a further section.

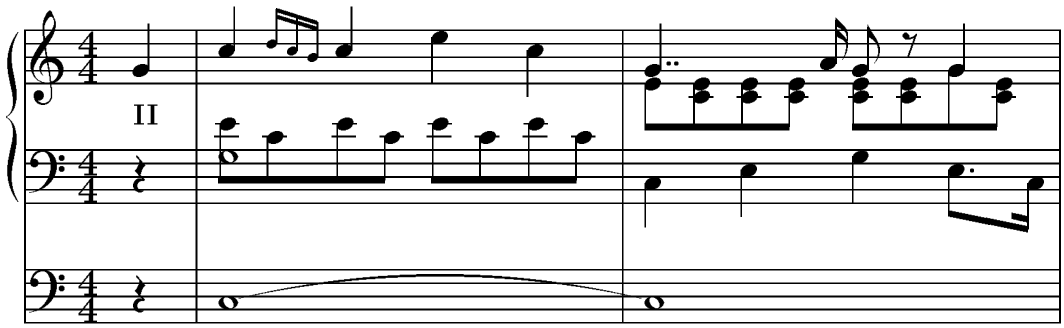
2.18.1 Cadenzas and explicit ornaments

Ornaments and *cadenzas* usually need to be written using smaller notes[28]. This can be done everywhere by stating `\smallnotesize` or `\tinynotesize`. Normal note size is restored by `\normalnotesize`.

[27]Its logic is similar to plain TEX's `\vadjust` command.

[28]This is independent of the staff size.

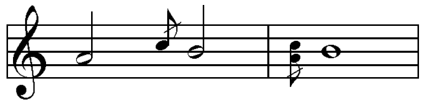
These macros only have a local scope. Thus, if these macros are invoked outside the `\notes...\enotes` pair, the change is valid for the rest of the piece unless explicitely modified but, if they are invoked inside, their effect is local to the current staff of the current `\notes...\enotes` pair. As an example, the following excerpt (beginning of the Aria of the "Creation" by Joseph HAYDN)



can be coded as:

```
\instrumentnumber{2}
\generalmeter{\meterfrac44}
\setstaffs2{2}
\setclef2{\bass}
\setclef1{\bass}
\startbarno=0
\startextract
\NOtes\qp&\zmidstaff{\bf II}\qp|\qu g\en
% mesure 1
\bar
\Notes\itieu2J\wh J&\zw N\ibl0c0\qb0e|\qu j\en
\notes&\ibl0c0\qb0c|\multnoteskip\tinyvalue\tinynotesize
  \Ibbu1ki2\qb1{kj}\tqh1i\en
\Notes&\qb0e\tbl0\qb0c|\qu j\en
\Notes&\ibl0c0\qb0{ece}\tbl0\qb0c|\ql l\sk\ql j\en
% mesure 2
\bar\Notes\ttie2\wh J&\ql J\sk\ql L|\zqupp g\qbl1e0%
  \zq c\qb1e\zq c\qb1e\zq c\tbl1\zqb1e\en
\notes&|\sk\ccu h\en
\Notes&\ql N\sk\ibl0L{-4}\qbp0L|\ibl1e0\zq c\zqb1e\cu g%
  \zq c\zqb1e\raise\Interligne\ds\zqu g\qb1g\en
\notes&\sk\tbbl0\tbl0\qb0J|\tbl1\zq c\qb1e\en
\endextract
```

2.18.2 Grace notes

Grace notes are a special case of small and tiny notes: the difference is that they are always coded as eighth notes with an oblique bar over the flag. To perform this, special variants of `\cu` and `\cl` have been provided, namely `\grcu` and `\grcl`, with the only difference that the flag has been slashed. Using this together with the note reduction macro, grace notes (optionally chord grace notes) can be easily coded:

The previous example was coded as:

```
\startextract
\NOTes\hu h\enotes
\notes\multnoteskip\smallvalue\smallnotesize\grcu j\enotes
\NOTes\hu i\enotes
\bar
\notes\multnoteskip\tinyvalue\tinynotesize\zq h\grcl j\enotes
\NOTEs\wh i\enotes
\endextract
```

2.18.3 Other note shapes

The classical note heads given above — namely , and — can be replaced with less classical note heads, for example to code special *violin harmonic notes* or *percussion music*. See an example in 2.21.2.3.

At present time, alternate available note heads can be found in the extension library, see 2.26.6, 2.26.14, 2.26.10 and 2.26.12.

2.19 Staff size

2.19.1 Moving from 20pt to 16pt, 24pt or 29pt staff sizes and conversely

You also want to write some parts of your score in 20pt staff size and others in 16pt or 24pt or 28.8pt staff size, namely for distinct parts of pieces. Changing the general staff size is done by saying:

`\smallmusicsize`, or `\normalmusicsize`, or `\largemusicsize`, or `\Largemusicsize`.
respectively.

2.19.2 Changing staff size for certain instruments

Regardless of the general choice of `\smallmusicsize` or `\normalmusicsize`, it is now possible to assign certain instruments – not separate staffs belonging to a same instrument – to have narrower or larger *staff size*, i.e. narrower or larger *staff spacing*. This is done using

`\setsize` *n*{*size*}

where *n* is the number of the instrument considered. Five different *sizes* are available as standards: `\normalvalue`, which is the default, `\smallvalue` (0.80 times narrower), `\tinyvalue` (0.64 times narrower), `\largevalue` (1.2 times larger) and `\Largevalue` (1.44 times larger). But, if you know perfectly what you are doing, you can also say:

`\setsize 3{2.0}`

which will provide the third instrument with staffs of spacing twice the standard value. In that case, MusiXTEX will use the nearest symbol size available, but this is likely to be rather ugly.

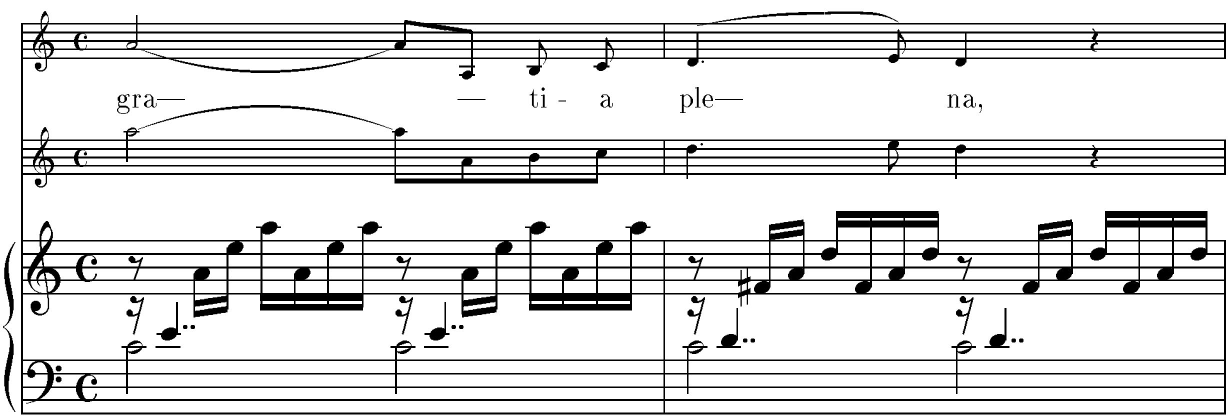
An alternate and equivalent coding — a remnant from MusicTEX — could be:

`\def\staffspacingiii{2.0}`

`\setsize` must be invoked before the starting command `\startpiece`, since this statement does not only change the vertical spacing between staff lines, but it also changes the size of the key, accidental and note symbols to fit the modified staff line spacing.

However, it must be emphasized that non-standard staff spacings use the nearest available font size. This make no serious problem for notes and accidentals which may be slightly too small or too big, but clefs and rests may appear to be shifted upwards or downwards. This trouble is progressively removed, but it requires changing the `musix*` fonts, with some possible characters forgotten... or the fact that user uses a font older than T.60 version.

As an example, we give two bars of the *Ave Maria* by Charles GOUNOD, based on the first prelude of Johann-Sebastian Bach's *Well Tempered Clavier* (transcription for organ, violin and voice, thanks to Markus VEITTES):



This example was coded as:

```
\def\oct{\advance\transpose by 7}
\def\liftqs#1{\raise#1\Interligne\qs}
\parindent0pt
\sepbarrules
\instrumentnumber{3}
\generalmeter{\meterC}
\setinterinstrument2{3\Interligne}
\setsize3\tinyvalue
\setsize2\tinyvalue
\setclef1\bass
\setstaffs1{2}
\startpiece\addspace\afterruleskip
%Takt 9
\notes\zhl c\liftqs6\qupp e|\ds&\oct
  \itieu5h\hl h&\tx ~~~gra---*\itied4h\hu h\enotes
\notes|\ibbl0j3\qb0h\tqb0l\enotes
\notes|\ibbl1k0\qb1{ohl}\tqb1o\enotes
\notes\zhl c\liftqs6\qupp e|\ds&\oct
  \ttie5\ibl4c0\qb4h&\ttie4\ibu5g{-3}\qb5h\enotes
\notes|\ibbl0j3\qb0h\tqb0l&\oct\qb4a&\tx ---*\tqh5a\enotes
\notes|\ibbl1k0\qb1o\qb1h&\oct\qb4b&\tx ~~ti~-*\cu b\enotes
\notes|\qb1l\tqb1o&\oct\tqb4c&\tx a*\cu c\enotes
\bar
%Takt 10
```

```
\notes\zhl c\liftqs6\qupp d|\ds&\oct
  \qlp d&\tx ~~~ple---*\ibsluru4e\qup d\enotes
\notes|\ibbu1g3\bigaccid\qb1{^f}\tqh1h\enotes
\notes|\ibbu2i0\qb2k\qb2f\enotes
\notes|\qb2h\tqh2k&\oct\cl e&\curve222\tubslur4f\cu e\enotes
\notes\zhl c\liftqs6\qupp d|\ds&\oct\ql d&\tx na,*\qu d\enotes
\notes|\ibbu1g3\qb1f\tqh1h\enotes
\notes|\ibbu2i0\qb2{kfh}\tqh2k&\qp&\qp\enotes
\endpiece
```

2.20 Layout parameters

Most layout parameters are set by MusiXTEX to reasonable default values. However, sophisticated scores[29] may need more place below the lowest staff, between staves, etc. We give below a short list of the most significant parameters.

2.20.1 List of layout parameters

REMARK: the mention "(NOT to be changed)" does not mean that this parameter cannot be changed, but that it should not be modified directly, e.g. by saying something like `\Interligne=14pt`. *In other words, changing these parameters must be performed using more comprehensive macros which not only update them but also perform some other compulsory related changes.*

`\Interligne` : vertical distance between the base of staff lines of the current instrument, taking no account of a possible specification of `\setsize`*n* (*NOT to be changed*).

`\internote` : the vertical spacing of contiguous notes of the current instrument, taking account of a possible specification of `\setsize`*n* (*NOT to be changed*).

`\Internote` : the vertical spacing of contiguous notes of the instrument(s) whose `\setsize`*n* has the *default value* of one (`\normalvalue`), i.e. the half of `\Interligne` (*NOT to be changed*)

`\staffbotmarg` : margin below the first staff of the lowest instrument. Changes are recognized at the next system and default is `3\Interligne`.

`\stafftopmarg` : margin above the upper staff of the upper instrument. Changes are recognized at the next system and default is `3\Interligne`.

`\interbeam` : vertical distance between beams. (*NOT to be changed*).

`\interportee` : the distance between the bottom of one staff and the bottom of the next one. It is set to `2\interstaff\internote` at the next system. Therefore, trying to change `\interportee` will have no effect.

`\interinstrument` : the additional vertical distance between two different instruments. This means that the distance between the upper staff of the previous instrument and the lowest line of the current instrument is equal to `\interportee+\interinstrument`. This value is normally zero, but it helps putting additional space between distinct instruments for

[29]To our knowledge, the most complicated scores are those written for the piano, during the romantic and post-romantic periods.

the sake of clarity. This is a general dimension register which holds for each of the vertical spaces between instruments, except above the upper one, in which case this interval is irrelevant. As usual in TEX, it can be set using a command such as

`\interinstrument=10pt`

or

`\interinstrument=6\internote`

However, this general parameter can be overridden for the space above a specific instrument. For example (see the example `angescao.tex`) one can state:

`\setinterinstrument` n`{1\Interligne}`

to force an additional spacing of one `\Interligne` above instrument n, whatever the value of `\interinstrument`. This feature can usefully be used to have more space before instruments representing *voices*, in order to have enough place to put *lyrics* without assigning these lyrics a zero staff specific instrument (useful to avoid having too many declared instruments in a choir score).

The `\setinterinstrument` is identical to the ancient — and still working — MusicTEX command:

`\def\interinstrumentiii{5\interligne}`

to add 5`\interligne` above the 3rd instrument.

`\systemheight :` the distance from the bottom of the lowest staff to the top of the highest staff of the upper instrument. This is the height of the vertical bars (single, double, repeats, etc.) (*NOT to be changed*).

In addition, when handling notes of a given staff of a given instrument, the following dimensions are available (note these are not true registers, but *equivalenced symbols* through a `\def`):

- `\altplancher` : the altitude of the lowest line of the lowest instrument (*NOT to be changed*).
- `\altitude` : the altitude of the lowest line of the lowest staff of the current instrument (*NOT to be changed*).
- `\altportee` : the altitude of the lowest line of the current staff (*NOT to be changed*).
- `\stemfactor` : a parameter defining the size of half, quarter and hooked eighth notes stems. Normally a stem has the length of one octave, i.e. 3.5`\Interligne`. However, this is not valid for small size notes and, therefore, the stem size is related to the `\interbeam` dimensions which, in turn, is *normally* equal to 0.75 `\Interligne`. Thus the normal value of `\stemfactor` is 4.66, but it can be shortened for any purpose by saying, for example:

  `\stemlength{3.5}`

2.20.2 Changing layout parameters

Most of these values can be changed, but only between the end of the previous system and the beginning of the next one. This can be inserted between a `\stoppiece` (or a `\endpiece`) and a `\contpiece` (or a `\startpiece`), but it is wiser to say, for example:

`\def\atnextline{\stafftopmarg=5\Interligne}`

The user may prefer to change `\staffbotmarg` or to feed `\interstaff` with a given integer number, but this can be done only between a `\stoppiece` (or a `\endpiece`) and a `\contpiece` (or a `\startpiece`), but it is wiser to use `\atnextline` as previously.

It is also wise to use `\atnextline` to change the number of instruments, the staff spacings, the number of staffs at the next line... provided that the coding of the notes *resists* an unexpected line change executing the `\atnextline`.

2.20.3 Changing the staff distance within systems

The vertical distance of instruments with more than one staff can be changed, either outside `\startpiece`...`\endpiece` or using `\atnextline`, by saying:

`\interstaff{`n`}`

Then the spacing between the staffs of the instruments will be n`\internote`. Note that `\interstaff` applies to all the instruments, but each distinct instrument may have a different `\internote` (see 2.19.2).

Note that this command is equivalent to an existing remnant of MusicTEX, namely:

`\def\interfacteur{`n`}`

The distance between two different instruments can be increased by saying

`\interinstrument=`*any TEX-dimension*

for all additional inter-instrument distances, or

`\setinterinstrument` n`{`*any TEX-dimension*`}`

for single distances.

2.20.4 Changing the number of lines in staffs

Unless explicitely specified, staffs consist of five lines, in accordance to the normal way of coding music scores. However, exceptions might be preferred when using MusiXTEX :

- *gregorian music* is often written using staff of four lines instead of five[30],
- *percussion music* (e.g. drums, triangle) needs one, two, three or five lines, since the pitch cannot change but the number of percussion instruments.
- *guitar tablature* needs six lines. Although *real* guitar players hate guitar tablature, for beginners they are a big help. Not ready yet. The lines are, but not the additional macros.
- Early baroque music sometimes uses staffs with 6, 7, or 8 lines.

Therefore MusiXTEX allows for choosing the number of lines of the staffs of an instrument. This is done by `\setlines` n`{`N`}` — where n is the number of the wanted instrument as usual — and N the number of lines of the specific staff. Allowed numbers are up to 9 lines. For example:

`\setlines2{4}`

will make the instrument number 2 to have staffs of four lines, that is, fit for gregorian music.

2.20.5 Resetting normal layout parameters

Except the general size which has to be explicitely changed if needed using `\smallmusicsize`, `\normalmusicsize`, `\largemusicsize`, or `\Largemusicsize`, all layout registers are reset to default values by `\resetlayout`, which are explicitly:

[30] German: *Quadratnotation*.

- `\staffbotmarg` and `\stafftopmarg` are set to `3\Interligne`.
- `\interstaff` is set to `9`.
- The number of lines of all instruments are reset to 5.
- All clef symbols are standard clef symbols.

Besides, saying `\begin{music}` invokes `\resetlayout`.

2.20.6 Typesetting one-line excerpts rather than large scores

Very often, what is wanted is not to typeset a large comprehensive score of several lines and pages, but an excerpt of one or two bars, preferably centered such as the various examples of this manual. This can be done simply by replacing `\startpiece` with `\startextract` and `\endpiece` or `\stoppiece` with `\endextract`. It is sometimes useful to get more than one excerpt in one line, this can be done with redefining of `\extractline`.

With saying:

```
\let\extractline\hbox
```

you can put more than one example in one line. Helpful for getting *musical footnotes*, which explain the playing of special trills or give alternative ways of playing, is the redefining to:

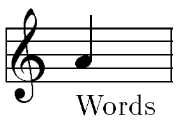
```
\let\extractline\leftline
```

Besides, all changes inbetween `\begin{music}`...`\end{music}` are local, means not global.

If you want to terminate it without a bar, you code `\zendextract` which acts like `\zstoppiece`.

2.20.7 *Lyrics

While they pose nearly no problem with MusicTEX which works using the glue facility of TEX, lyrics raise several serious difficulties with MusiXTEX. The reason for that is that lyrics usually consist of *text* which in turn consists of characters whose width is definitely not stretchable nor shrinkable proportionally to `\noteskip` or `\elemskip`.

Thus the width of a sequence of notes like

may be either the width of "`Words` " with the current font, or it may be `1\elemskip`, whatever the largest. In the first case, the width is fixed (see `\hardspace`), in the second its width is not the same at MusiXTEX's first and second passes. But the unfortunate situation is that MusiXTEX *must* know whether this feature has a *hard* (not scalable) or a *scalable* width *at the first pass*, while the inequality is solved only at the second.

This means that, in addition to the three pass system, the user must run trial and errors to find out whether some lyrics are wider than the final note spacing, or smaller. In that case a wrong decision might lead to questionable "`Underfull hbox`"-es filled with ugly glue, or it may result in awful "`Overfull hbox`" with its heavy bar punishment and possible over writings.

However, before discussing this problem, we first describe the various ways of inserting lyrics under — or above — a staff.

2.20.7.1 Getting enough vertical space for lyrics

Since songs are usually equivalent to a one-staff instrument (possibly with several voices) the recommended solution consists in adjusting the distance between instruments using either `\interinstrument=`*any TeX-dimension* to give more place below all instruments or using `\setinterinstrument` to make more space above. Note that `\setinterinstrument` defines spacing above and not below an instrument. Since lyrics are usually set below the staff, the first argument of a `\setinterinstrument` should be the song instrument number *minus one*.

In the case of a single staff tune, or if the song instrument is the lowest one, then additional place can be provided using `\staffbotmarg`.

2.20.7.2 Posting lyrics

Lyrics can be introduced in several ways.

1. One archaic solution could be to define a special instrument for the lyrics text, without effective staff, namely using the command `\setstaffs` n{p} described in 2.2.2. The inconvenience of this solution is that it increases the number of instruments, which is limited to 6 in standard MusiXTeX, although it can be increased to 9 using `musixadd.tex` and 12 with `musixmad.tex`.

   Therefore, better solutions consist is posting the lyrics in the same instrument as the tune, provided that inter-instrument spacing has been adapted.

2. Another simple obvious solution consists in using the command `\zcharnote` to post the text at any position (computed in `\internote`s) with respect to the lower line of the current staff. The pitch should be usually negative to have the text below the staff. It could also be specified like a note pitch, for example "`a`" if the song is in G-clef. The drawback is that the pitch has to be repeated for each text of each note, which is tideous unless a special macro has been defined.

3. Rather than `\zcharnote` and other `\?charnote` commands, one should rather use `\zchar`, `\cchar` (centered) or `\lchar` (expanded to left) which only allow absolute numbers (internally multiplied by `\internote`). These command have the same drawback as above, namely the vertical position has to be repeated each time.

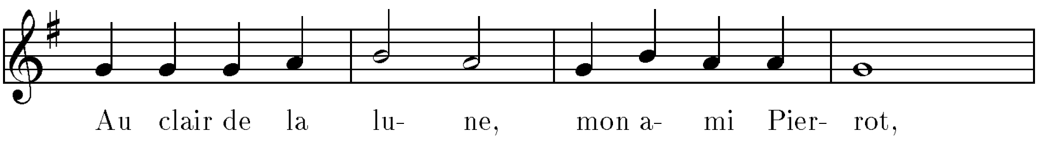
4. Of easier use are the commands `\zsong` (right of the note), `\lsong` (left) and `\csong` (centered) which post the lyrics at the lower staff line *minus* the previous `\interinstrument` n or the `\staffbotmarg` quantity. These commands only have one argument, namely the lyrics text:

   `\zsong{`*text*`}` `\lsong{`*text*`}` `\csong{`*text*`}`

   Depending on the values of the inter-instrument spacings and margins, the resulting vertical position might be inappropriate. Then it can be changed for any specific n-th instrument until further change using

   `\setsongraise` n{*any TeX-dimension*}

   As an example, the following French song



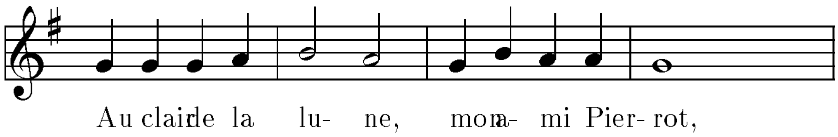
was coded as:

```
\generalsignature{1}
\startextract
\geometricskipscale
\NOtes\zsong{Au }\qu g\en
\NOtes\zsong{clair }\qu g\en
\NOtes\zsong{de }\qu g\en
\NOtes\zsong{la }\qu h\en
\bar
\NOTes\zsong{lu- }\hu i\en
\NOTes\zsong{ne, }\hu h\en
\bar
\NOtes\zsong{mon }\qu g\en
\NOtes\zsong{a- }\qu i\en
\NOtes\zsong{mi }\qu h\en
\NOtes\zsong{Pier- }\qu h\en
\bar
\NOTes\zsong{rot, }\wh g\sk\en
\endextract
```

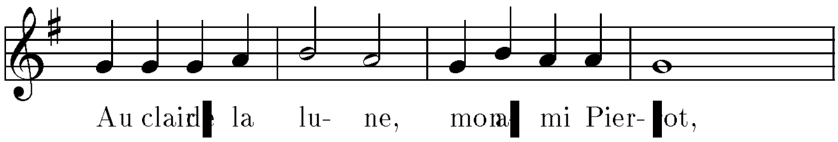
You can also use the macros from `musixcho.tex`, which result in the same vertical positionning as `\zsong` and other `\?song` do, but the text of the lyrics are better justified around the related note.

2.20.7.3 Handling lyrics width versus scalable note spacings

If one uses either `\zsong` or `\csong` or `\lsong` without caution, one will probably get awful things such as:



which is the same example as above, with all spacings divided by 1.41. Although locating "overfull" lyrics seems easy, it might be wiser to force their visibility, which is done replacing `\zsong` with `\hsong` which encloses the hazardous text in a `\hbox` of width `\noteskip`, whose overfilling is diagnosed as usual by Plain TEX or (especially in LATEX) by means of the `\overfullrule=...` command:



Alternate versions of `\hsong` are `\dhsong` which has a fixed length of `2\noteskip` and `\thsong` whose fixed length is `3\noteskip`. These are useful when the text is set below (or above) a collective coding of two or three notes.

Once one has diagnosed — in the tentative final layout — which lyrics lead to overfull texts, several corrections may be tried:

1. Increase the `\mulooseness` to have wider note spacing. This may work, but the inconvenience is that all notes would be stretched, and not only the faulty notes.

2. Increase specific note spacings, replacing for example `\Notes` with `\NOtes` or `\Notesp`. This is the easiest solution which may require several trials and errors since an increase of a few notes in a system may lead to a small shrinking of the others, to keep the total length constant.

3. Decide that some notes — only "some" notes, not all notes of the score, neither all the notes of a given system — would have a *hard spacing* and not a scalable spacing. This is done using the command

   `\hardnotes` *any hard TEX dimension* `\notes` *normal note specif.* `\en`

   which ends the `\notes...\en` with a *hard spacing* (see `\hardspace` in 2.8, p. 34) rather than a scalable spacing.

   A more practical variant has been provided especially for lyrics, namely

   `\hardlyrics` *any text* `\notes` *normal note specif.* `\en`

   which computes the width of "any text" by putting it in a `\hbox` and reverting to `\hardnotes` to perform the remainder of the task. In order to save repetitions, the text of the lyrics which has been used to compute the hard note spacing can be retrieved under the command name `\thelyrics`. `\hardlyrics` must be followed by `\notes` (with the lyrics text inbetween), not by `\Notes` nor `\NOtes`, etc.

   Besides, since moving the lyrics text between `\hardlyrics` and `\notes`[31] together with replacing it with `\thelyrics` at the initial location is a rather tedious operation with text editors, another command has been provided, namely `\softlyrics`:

   `\softlyrics{` *any text* `}`

   defines `\thelyrics` to be " *any text* ", so that going from `\hardlyrics` to the default scalable behaviour of `\noteskip` and inversely can easily be done by changing `\hard` into `\soft` in the source text, with optional change of the number of uppercase letters in `\notes`.

   IMPORTANT: since `\hardlyrics` *computes* `\noteskip`, *its argument (before* `\notes`*) may not refer to* `\noteskip`*... which is not already computed.*

2.20.7.4 A more comprehensive example of lyrics

Here is an example taken from the French catholic liturgy (a French translation of the traditional *Gloria in excelsis Deo*). We give first the adapted source of a part of the score:

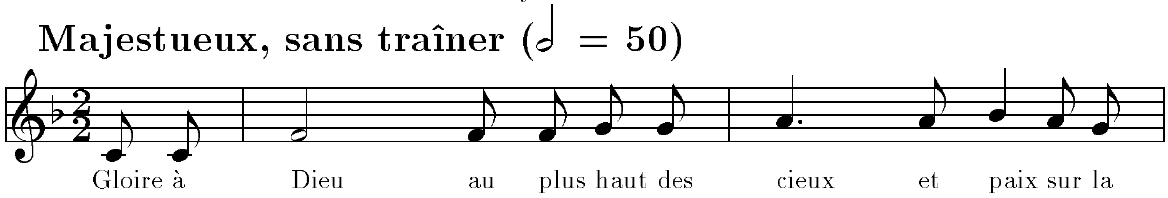
```
\overfullrule 3pt
\instrumentnumber{1}
\setstaffs11
%
\generalsignature{-1}\relax % one flat
\generalmeter{\meterfrac{2}{2}}
%
\staffbotmarg=5\Interligne\stafftopmarg=1.5\Interligne
\startpiece
\setsongraise1{2\Interligne}%
```

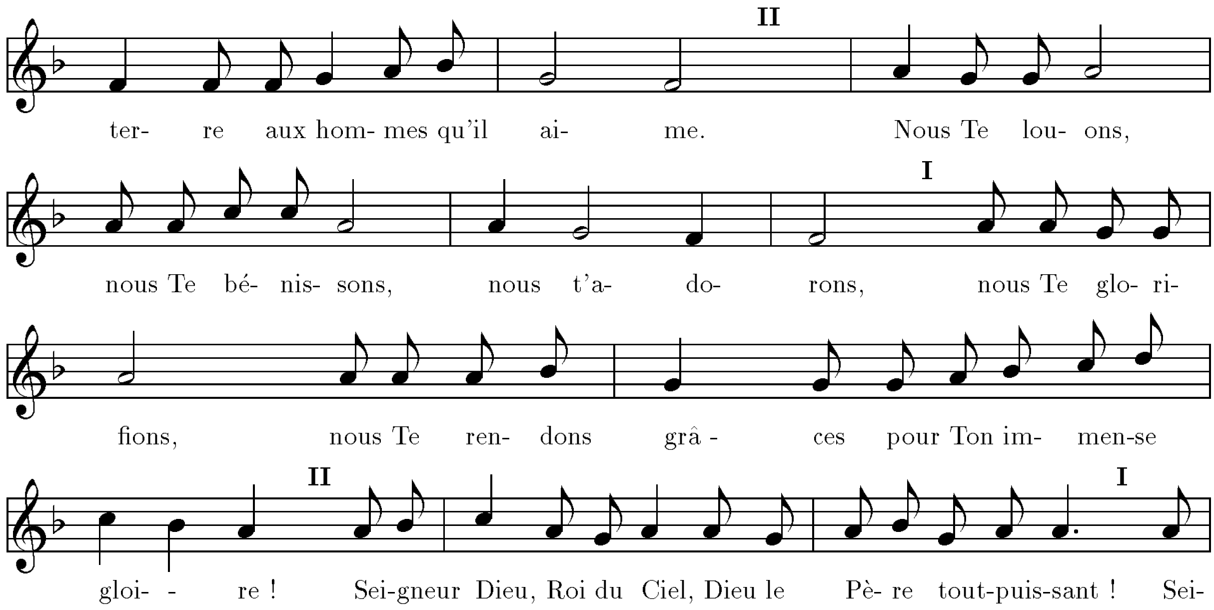
[31] `\notes`, neither `\Notes` nor `\NOtes` nor `\NOTes` etc.

```
\znotes\uptext{\kern -9mm\raise 9pt\hbox{\bigtype
  Majestueux, sans tra\^\i ner \rm(\metron{\hu}{50})}}\enotes
\hardlyrics {\kern-5pt Gloire }\notes\hsong{\thelyrics }\cu c\enotes
\notes\hsong{\`a }\cu c\enotes
\barre
\NOTes\hsong{Dieu~}\hu f\enotes
\notes\hsong{au}\cu f\enotes
\hardlyrics{plus~}\notes\hsong{\thelyrics }\cu f\enotes
\hardlyrics{haut~}\notes\hsong{\thelyrics }\cu g\enotes
\notes\hsong{des}\cu g\enotes
\barre
\NOtes\hsong{cieux~}\qup h\enotes
\notes\hsong{et~}\cu h\enotes
\hardlyrics{paix~}\notes\hsong{\thelyrics }\qu i\enotes
\hardlyrics{sur~}\notes\hsong{\thelyrics }\cu h\enotes
\notes\hsong{la~}\cu g\enotes
\barre
\Notes\hsong{ter-}\qu f\enotes
\notes\hsong{re~}\cu f\enotes
\hardlyrics{aux~}\notes\hsong{\thelyrics}\cu f\enotes
\hardlyrics{hom- }\notes\hsong{\thelyrics}\qu g\enotes
\hardlyrics{mes~}\notes\hsong{\thelyrics }\cu h\enotes
\hardlyrics{qu'il~}\notes\hsong{\thelyrics }\cu i\enotes
\barre
\NOtes\hsong{ai-}\hu g\enotes
\Notes\hsong{me.}\hu f\enotes
\Notes\uptext{\bf II }\enotes
\barre
\hardlyrics{Nous~}\notes\hsong{\thelyrics}\qu h\enotes
\notes\hsong{Te~}\cu g\enotes
\notes\hsong{lou-}\cu g\enotes
\NOtes\hsong{ons,}\hu h\enotes
\barre
\hardlyrics{nous~}\notes\hsong{\thelyrics }\cu h\enotes
\notes\hsong{Te~}\cu h\enotes
\notes\hsong{b\'e-}\cu j\enotes
\notes\hsong{nis-}\cu j\enotes
\NOtes\hsong{sons,}\hu h\enotes
\barre
\Notes\hsong{nous~}\qu h\enotes
\NOtes\hsong{t'a-}\hu g\enotes
\Notes\hsong{do-}\qu f\enotes
\barre
\NOtes\hsong{rons,~}\hu f\enotes
\notes\uptext{\bf I}\enotes
\hardlyrics{nous~}\notes\hsong{\thelyrics }\cu h\enotes
\notes\hsong{Te~}\cu h\enotes
\notes\hsong{glo-}\cu g\enotes
\notes\hsong{ri-}\cu g\enotes
\barre
```

```
\NOTEs\hsong{fions,~}\hu h\enotes
\hardlyrics{\kern -4pt nous~}\notes\hsong\thelyrics\cu h\enotes
\notes\hsong{Te~}\cu h\enotes
\notes\hsong{ren-}\cu h\enotes
\notes\zsong{dons~}\cu i\enotes
\barre
\NOtes\hsong{gr\^a~-}\qu g\enotes
\notes\hsong{ces~}\cu g\enotes
\hardlyrics{pour~}\notes\hsong\thelyrics\cu g\enotes
\hardlyrics{Ton~}\notes\hsong\thelyrics\cu h\enotes
\notes\hsong{im-}\cu i\enotes
\hardlyrics{men-}\notes\hsong\thelyrics\cu j\enotes
\notes\zsong{se~}\cu k\enotes
\barre
\Notes\hsong{gloi-}\ql j\enotes
\Notes\hsong{-}\ql i\enotes
\Notes\hsong{re~!}\qu h\enotes
\notes\uptext{\bf II}\enotes
\notes\hardlyrics{Sei-}\hsong\thelyrics\cu h\enotes
\notes\zsong{gneur~}\cu i\enotes
\barre
\hardlyrics{Dieu,~}\notes\hsong\thelyrics\qu j\enotes
\hardlyrics{Roi~}\notes\hsong\thelyrics\cu h\enotes
\notes\hsong{du~}\cu g\enotes
\Notes\hsong{Ciel,~}\qu h\enotes
\hardlyrics{Dieu~}\notes\hsong\thelyrics\cu h\enotes
\notes\zsong{le~}\cu g\enotes
\barre
\hardlyrics{P\'e-~}\notes\thelyrics\hsong\cu h\enotes
\notes\hsong{re~}\cu i\enotes
\hardlyrics{tout-}\notes\hsong\thelyrics\cu g\enotes
\hardlyrics{puis-}\notes\hsong\thelyrics\cu h\enotes
\hardlyrics{sant~!}\notes\hsong\thelyrics\qup h\enotes
\notes\uptext{\bf I}\enotes
\notes\zsong{Sei-}\cu h\enotes
\mulooseness 2%
\stoppiece
```

The result clearly shows the inequality between note spacings, some of them forced by lyrics length, some others related to the value of `\elemskip` choosen by `musixflx` according to the given value of `\mulooseness`. Note that `\zsong` still has been used for lyrics followed by a bar, since their extension across the bar is usually harmless.





The complete score of this carol — with slightly different layout due to different page size — is given in example `glorias.tex` and in `gloriab.tex`, the latter exhibiting not only the song tune but also the organ accompaniment (both require `musixsty.tex`).

2.21 Abnormal music coding

2.21.1 Gregorian chant: staffs and clefs

Gregorian chant is often coded using four line staffs (see sections 2.20.4 and 2.20.4) and using special notes (called *neumes*) which are described in section 2.26.10. But the gregorian chant also needs special clefs which are in fact the ancestors of the modern ones. The *gregorian clefs* can be invoked instead of the modern ones by:

`\setaltoclefsymbol3\gregorianCclef`

`\setbassclefsymbol3\gregorianFclef`

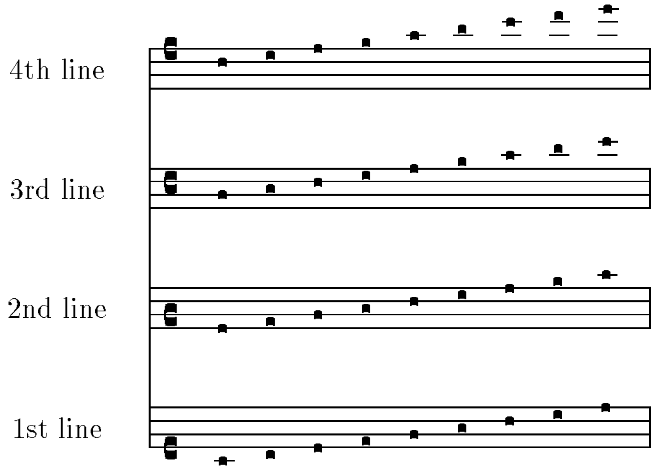
will cause the instrument number 3 to exhibit the chosen gregorian clefs. The standard clefs can be restored for every instrument with `\resetclefsymbols`.

It is to be emphasized that, since version T.40 of MusiXTEX, one should specify whether one wants to change the bass clef symbol or the alto clef symbol (also the treble clef symbol, but there is no treble clef in gregorian chant). The reason is that MusiXTEX (as well as MusicTEX) selects and raises differently the F, G and C clefs according to the arguments of the `\setclef` command. Therefore, if one had substituted any F clef symbol while saying `\setclef1{1000}`, then an F clef would duly appear on the staff, but it would have been set at the position of an alto clef, thus seriously misleading the musician.

However, for compatibility with what was previously choosen[32], the `\setclefsymbol` still exists, but it substitutes the given second argument *to all clef symbols* of that instrument, regardless of the actual musical meaning of this symbol. `\setclefsymbol` is therefore discouraged, unless one changes the design of the clef symbol of an instrument for the whole of the score.

As an example, the same gregorian scale has been written with a gregorian C clef on all four lines of the staff:

[32] By A. EGLER...



The coding was:

```
 \instrumentnumber{4}
 \setname1{1st line} \setname2{2nd line} \setname3{3rd line} \setname4{4th line}
 \setlines1{4}\setlines2{4}\setlines3{4}\setlines4{4}
 \sepbarrules
 \generalmeter{\empty}
 \setclef1{1000} \setclef2{2000} \setclef3{3000} \setclef4{4000}
 \setaltoclefsymbol1\gregorianCclef
 \setaltoclefsymbol2\gregorianCclef
 \setaltoclefsymbol3\gregorianCclef
 \setaltoclefsymbol4\gregorianCclef
\startextract
\Notes\squ{abcdefghi}&\squ{abcdefghi}&\squ{abcdefghi}&\squ{abcdefghi}&\enotes
\endextract
```

2.21.2 Music score without clefs or with special clefs

2.21.2.1 Empty clefs

Regardless of the number of lines of the staffs, an instrument may have no clefs, e.g. for *percussion music* but also for any weird purpose. This is done by declaring the following item:

`\setclefsymbol` n`\empty`

Normal symbols for these clefs can be restored by:

`\resetclefsymbols`

2.21.2.2 Octave clefs

Octave treble clefs and octave bass clefs are provided. They can replace the usual treble and bass clefs by saying:

`\setclefsymbol` n`\bassoct`

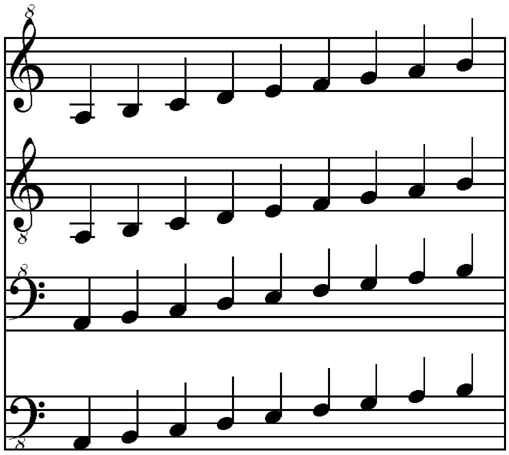
`\setclefsymbol` n`\trebleoct`

for upper octaviation, and

`\setclefsymbol` n`\basslowoct`

`\setclefsymbol` n`\treblelowoct`

for lower octaviation.



REMARKS:

- *The* `\setclefsymbol`*,* `\settrebleclefsymbol`*,* `\setaltoclefsymbol` *and* `\setbassclefsymbol` *commands only change the symbol posted, not the altitude of the note head. Thus, if you say* `\setclef1\bass` *and then ——, your score will be definitely wrong. In the same way,* `\trebleoct` *and similar octave clefs will only change the symbol, but a note coded* `a` *wiull have the same vertical position, regardless of the clef symbol chosen.*

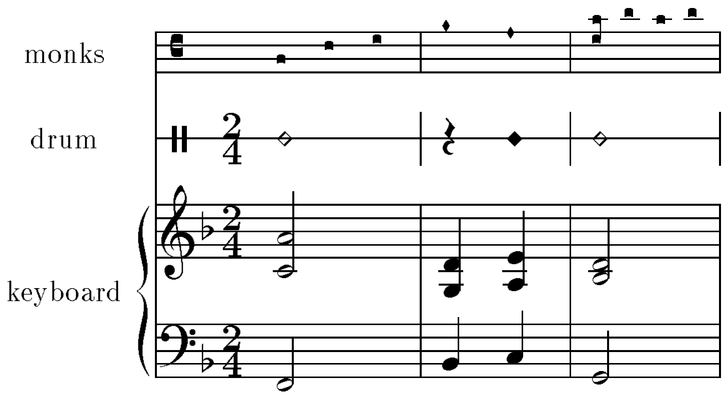
- *Changing clef symbols also affect clef changes within the score, but this clef changes will be posted by large symbols, not smaller clefs as usual. Anyway, it can be remarked that clef changes only concern scores with standard treble, alto and bass clefs...*

2.21.2.3 Drum clef

A special *drum clef* (two heavy vertical bars) can replace for the n-th intrument any of the standard clefs by saying:

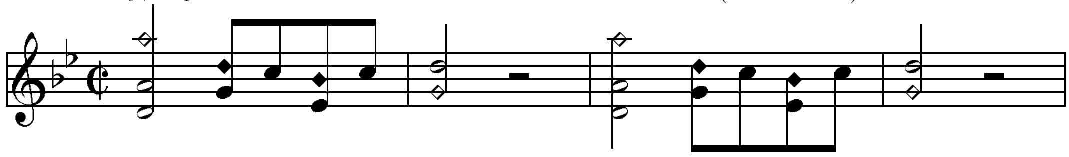
`\setclefsymbol` n`\drumclef`

However, since the vertical position of the clef depends on the previously stated clef, it is not wise to replace any clef symbol with the drum clef, but to have the instrument previously configured with the default violin clef, i.e. `{0000}`.

It is to be emphasised that these features are specific to one instrument — not one staff of a several staff instrument — so that some weird score for *monks*, *drum* and *electronic keyboard* such as

could be coded as follows, regardless this is relevant (see remark in 2.26.10, p. 86):

```
\parindent 19mm
\instrumentnumber{3}
\setname1{keyboard} \setname2{drum} \setname3{monks}
\setlines2{1}
\setlines3{4}
\setinterinstrument1{-2\Interligne}% less vertical space above
\setinterinstrument2{-2\Interligne}% and below the percussion
\sepbarrules
\setsign1{-1} % one flat at keyboard
\generalmeter{\meterfrac24}
\setmeter3{\empty}
\setclef3{\alto}
\setclef1{\bass}
\setstaffs1{2} % 2 staffs at keyboard
\setclefsymbol3{\gregorianCclef} % gregorian C clef at instrument 3
\setclefsymbol2{\drumclef}       % cancel G clef at instrument 2
\startextract
\Notes\hu F|\zh c\hu h&\dnq4&\squ{acd}\enotes\bar
\NOtes\qu I|\zq N\qu d&\qp&\diapunc f\enotes
\NOtes\qu J|\zq a\qu e&\ynq4&\diapunc f\enotes\bar
\notes\hu G|\zh b\hu d&\dnq4&\zsqu d\rsqu g\squ{hgh}\enotes
```

In the same way, a possible violin score with *harmonic notes* (see 2.18.3) could be:



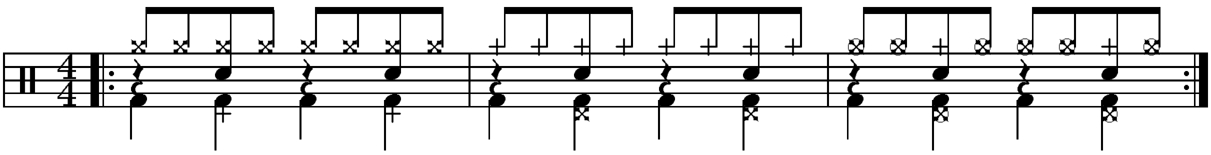
It was coded as follows:

```
\generalsignature{-2}
\generalmeter\allabreve
\startextract
\NOTes\dzq o\zh d\hu h\enotes
\Notes\ibu0k0\zq g\yqb0k\qb0j\zq e\yqb0i\tbu0\qb0j\enotes
\bar
\NOTes\dzq g\hu k\enotes
\NOTes\hpause\enotes
```

```
\bar
\NOTes\dzq o\zh d\hl h\enotes
\Notes\ibl0e0\zq g\yqb0k\qb0j\zq e\yqb0i\tbl0\qb0j\enotes
\bar
\NOTes\dzq g\hu k\enotes
\NOTes\hpause\enotes
\endextract
```

2.21.3 Usual percussion music

Besides single percussion scores usually written using one-line staffs, percussion music involving several instruments is often writen on five-line staffs with a *drum clef*, where the instruments are distinguished by the type of the note heads and the apparent pitch of the note on the staff. We give an example — kindly provided by Agusti Martín Domingo:



Its coding was:

```
\begin{music}
\instrumentnumber{1}
\generalmeter{\meterfrac44}
\setclefsymbol1\drumclef
\parindent0pt\startpiece
\leftrepeat
\Notes\zql f\rlap\soupir\ibu0m0\xqb0{nn}\enotes
\Notes\kzq d\zql f\zq j\xqb0n\tbu0\xqb0n\enotes
\Notes\zql f\rlap\soupir\ibu0m0\xqb0{nn}\enotes
\Notes\kzq d\zql f\zq j\xqb0n\tbu0\xqb0n\enotes
\bar
\Notes\zql f\rlap\soupir\ibu0m0\kqb0{nn}\enotes
\Notes\xzq d\zql f\zq j\kqb0n\tbu0\kqb0n\enotes
\Notes\zql f\rlap\soupir\ibu0m0\kqb0{nn}\enotes
\Notes\xzq d\zql f\zq j\kqb0n\tbu0\kqb0n\enotes
\bar
\Notes\zql f\rlap\soupir\ibu0m0\oxqb0{nn}\enotes
\Notes\oxzq d\zql f\zq j\kqb0n\tbu0\oxqb0n\enotes
\Notes\zql f\rlap\soupir\ibu0m0\oxqb0{nn}\enotes
\Notes\oxzq d\zql f\zq j\kqb0n\tbu0\oxqb0n\enotes
\setrightrepeat\endpiece
\end{music}
```

To use these different note heads, one must

- either include a specific percussion file namely `musixper.tex` after the usual `\input musixtex`;
- use special macro names to replace the usual elliptic black note head with other available note heads. Today only the most used note types are supported.

2.22 Writing your own macros: the `\catcode` problems

As seen before, the `\catcodes` of the `|` and `&` symbols are modified by MusiXTEX, in the range of the actual scores but not in the whole of the TEX source. Thus, if you define your own macros to make your writing easier, you are likely to invoke the `|` or `&` symbols in a part of text where their `\catcode`s are not correctly set. This may result typically in a diagnostic like:

```
! Misplaced alignment tab character &.
```

when you attempt, not to define, but to use your macro using the `&` symbol to change the instrument. Smart TEXers know that the `\catcode`s are attached to the characters *when they are input* and not when they are used; thus you must be sure that `|` and `&` have the correct MusiXTEX `\catcode` when the macro is defined, which may well occur outside the actual score.

It is also worth pointing out that the same problem may occur with other punctuation marks like "`<`", "`>`", "`^`", etc., if their `\catcode` has been changed by some other set of macros, like `french.sty`.

2.23 LATEX and MusiXTEX

2.23.1 The `musixtex.sty` style

As said before, the amount of memory and registers used by MusiXTEX makes it hardly compatible with LATEX. However, Nicolas BROUARD succeeded in building a `musixltx.tex` which is now included in the distribution. This is not recommended to make separate music scores. Its purpose is rather to provide a means of inserting short musical excerpts in books or articles written with LATEX. Then, the `\documentstyle` command should include `musixtex` in the options.

The LATEX style file `musixtex.sty` simply `\input`s the following files (in that very order):

- `musixtex.tex`
- `musixltx.tex`

In case of `TeX capacity exceeded...`, use a "BiglaTEX" (after checking there is no visible error in the source code).

2.23.2 Wide music in LATEX

Another difficulty appears with LATEX: internal LATEX macros handle the page size in a way which is not supposed to be changed within a given document. This means that text horizontal and vertical sizes are somewhat frozen so that one can hardly insert pieces of music of page size different from the size specified by the LATEX *style*. Although a `musixblx.tex` has been provided, which makes the *environment* `bigmusic` available. The main drawback is an unpredictable behaviour of top and bottom printouts, especially page numberings.

If the whole of a document has wide pages, it can be handled with the `a4wide` style option, or any derivate of it.

2.23.3 The `\catcode` problems

The `musixltx.tex` file merely overrides the `\catcode`s of the `|` and `&` symbols which are modified by MusiXTEX. To have access to these symbols when coding music, you should enclose the scores or excerpts within `\begin{music}` and `\end{music}`. But there is also another possibility, i.e. to say `\nextinstrument` instead of `&` and `\nextstaff` instead of `|`.

Another problem comes from the `french.sty` written by Bernard GAULLE which is the standard of the GUTenberg French association. This style changes many `\catcode`s which lead MusiXTEX to fail in many cases. Therefore the `\catcode`s of all are forced to the adequate value at `\startpiece`, `\startextract` and restored at their original value at `\endpiece`, `\stoppiece` and `\endextract`. This means that some facilities like the *guillemets* or the *tabulation* character are inhibited within music scores (possible problem with sophisticated *lyrics*) but perfectly available within the normal text.

Anyway, in case of emergency, one can invoke `\catcodesmusic` to establish the `\catcode`s at their value fit for music, and `\endcatcodesmusic` to reset them at their external value, for example those chosen by `french.sty`.

2.24 Implementation and restrictions

The macroinstruction file MusiXTEX contains approximately 4200 lines of code, that is 98 000 bytes approximately. This requires your score to be compiled by the most extended versions of TEX (65 000 words of working memory). It is therefore wise to say `\tracingsstats=2` in order to have an information about the memory used in each page. In desperate situations, we recommend using the "BigTEX" processors which, unfortunately, performed a great deal of disk input/outputs (on PCs with i286 processors) which made them awfully slow. Of course, using i386 or i486 or Pentium processors, this problem disappears with the specific version of `emTeX`, namely `tex386`.

In particular, the number of registers it uses and the amount of memory used by LATEX macros makes it doubtfully compatible with LATEX, unless using BigLATEX.

Other precautions are necessary: beware of end-of-line spaces; they corrupt layout and cause `underfull/overfull hbox` warnings during third pass. To avoid that, it is recommended to use `%` at the end of source lines.

2.25 From MusicTEX to MusiXTEX: `musixcpt`

This file is intended to run '**old**' input files with as less changes (read `musicdoc` **and** MusiXTEX's laws !) as possible. It slows down the speed and makes it impossible to use some[33] of MusiXTEX commands.

To run MusicTEX examples — not withstanding restrictions related to glue and "hard" spacings forbidden with MusiXTEX — just replace all previous `\input music*` by:

```
\input musixtex
\input musixcpt
```

and, optionally

```
\input musixsty
```

2.25.1 Compatibility restrictions

The `musixcpt.tex` file holds up compatibility to MusicTEX's sources, but there are some restrictions, which are caused either by missing glue in MusiXTEX — sometimes ugly, but gen-

[33] This was the will of one of the authors (A.E.) but we (we=D.T.) are now tending to remove any incompatibilities of `musixcpt` with plain MusiXTEX.

erally harmless — or by some hard spacings such as `\hbox`es containing lyrics exceeding the `\noteskip` dimension, and resulting in “overfull hbox[es]” that is, notes and symbols outside the score limits, with the heavy vertical bar as a standard TEX punishment.

The easiest way to avoid this consists in ending systematically *all*[34] input lines between `\startpiece...\stoppiece`[35] with a comment sign[36], namely a “`%`”.

Besides, one might want to have a unique source file to be compiled with both the MusicTEX format and the MusiXTEX format, and[37] taking advantage of some specific possibilities of MusiXTEX. The leads to a need of testing whether the source file is compiled with one format or the other. In fact this test is easy:

`\ifx\mxversion\undefined`

the specific MusicTEX code

`\else`

the specific MusiXTEX code, taking advantage of new features

`\fi`

Several examples are now included in MusicTEX's examples, for example `pacifiqn.tex`. Of course this would fail if the user inadvertently decided to define the command `\mxversion` in his source text.

2.25.2 Additional commands in musixcpt.tex

The tentative user looking in given examples might find commands not described in the MusiXTEX manual (this one). Most frequent are (in case of missing description, revert to MusicTEX's manual, namely `musicdoc.tex`, `musicdoc.dvi`, `musicdoc.ps`, musicdoc.lj, etc.):

- `\qh` : same as `\qb`, but was different with MusicTEX, since it distinguished notes hanging above (`\qb`) and below (`\qh`) the beam(s).
- `\debutmorceau` : nearly but not exactly the same as `\startpiece`.
- `\suspmorceau` : suspends the score at the end of the line, in order to permit inserting ordinary TEX commands. Similar to `\stoppiece` but performs line counting necessary then using MusicTEX's `\autolines` feature.
- `\reprmorceau` is synonym of `\contpiece`.
- `\lreprmorceau` is synonym of `\contpiece`.
- `\preprmorceau` ejects a newpage and performs `\contpiece`.
- `\finmorceau` is synonym of `\Stoppiece`, i.e. with heavy ending double bar..
- `\barre` : same as `\bar`.
- `\dsoupir` : same as `\ds`.
- `\qsoupir` : same as `\qs`.

[34] ALL lines, not “most of them”.
[35] or `\Stoppiece` or `\endpiece` or `\Endpiece`.
[36] For TEXperts: every line which does not end with a command
[37] This might be considered as an excessive requirement, but we (D.T.) think it is a good way of including MusiXTEX's new facilities...

- `\demisoupir` : same as `\ds`.
- `\temps` : was glue in MusicTEX, now equivalent to `\empty`.
- `\autolines` is fit for scores with bars of regular length: after `\startpiece`, you code the following macro:

  `\autolines` *tml*

  where t is the number of *elementary spacings* (the length of `\notes...\enotes`) in an average bar, m is the number of bars you wish in a line and l is the number of lines you wish in a page. This sets some parameters, namely `\maxbarsinline` and `\maxlinesinpage` which are simply used to count the bars (counter is `\barsinline`), optionally perform `\alaligne` or `\alapage` (counter is `\linesinpage`) instead of the normal `barre` or `bar`. You may freely alter the values of these parameters, once they have been established by `\autolines`. Moreover, you can still force line-breaking of page ejection using `\alaligne` or `\alapage` without problem since these macros actually reset the bar counters appropriately.

2.26 Extension Library

All following files are invoked with saying `\input` *filename*

2.26.1 musixadd

Increases the number of instruments, slurs and beams up to nine.

2.26.2 musixbm

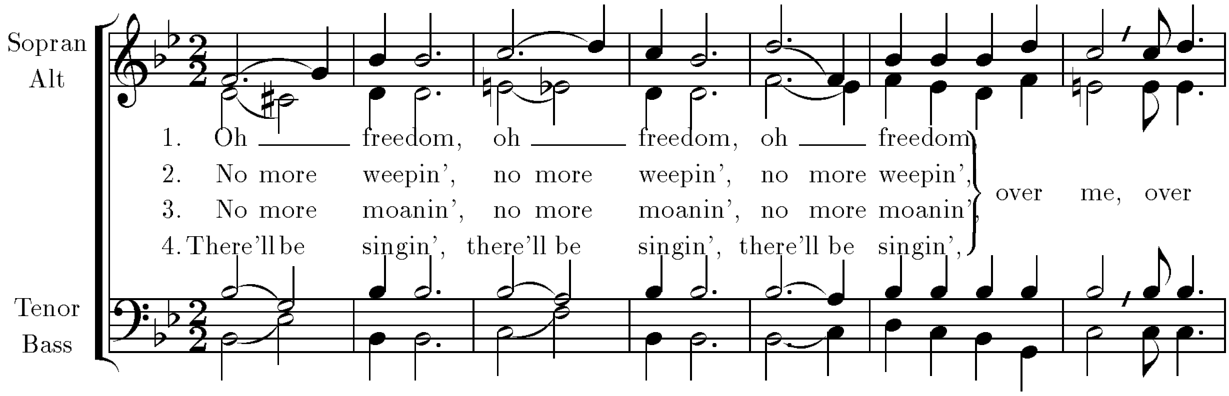
Provides 128th notes, either with hooks or with beams, namely the `\ibbbbbu`, `\ibbbbbl`, `\nbbbbbu`, `\nbbbbbl`, `\tbbbbbu`, `\tbbbbbl`, `\Ibbbbbu`, `\Ibbbbbl`, `\cccccu`, `\ccccca`, `\zcccccu`, `\cccccl` and `\zcccccl` commands.

2.26.3 musixbbm

Provides 256th notes, only with beams, namely the `\ibbbbbbu`, `\ibbbbbbl`, `\nbbbbbbu`, `\nbbbbbbl`, `\tbbbbbbu`, `\tbbbbbbl`, `\Ibbbbbbu` and `\Ibbbbbbl` commands.

2.26.4 *musixcho

These are some macros fit for chorus music. Provides following commands: `\biglbrace`, `\bigrbrace`, `\braceheight`, `\Dtx` and `\Drtx` for double text, `\Ttx` and `\Trtx` for triple text, `\Qtx` and `\Qrtx` for quadruple text.

`\tx{`*text*`}`, `\rtx{`*text*`}` for text of songs, with text extending to right instead centered. `\hf{`m`}` sets text continuation rule of width m `\noteskip`.

See `musixdoc.tex` source to get the coding of this long example.

Note that, since vertion T.89, multiple line texts are vertically justified with a `\ChroirStrut` macro defined as `\vphantom{\^Wgjpqy}` to eliminate zigzagging lyrics lines.

2.26.5 musixdat

Here is the command `\today` defined for several languages. The default is `\dateUSenglish`, but this can freely be changed at end of `musixdat.tex`. Also available are `\dategerman`, `\dateaustrian`, `\dateenglish` and `\datefrench`, which yields:

`\dateUSenglish`	January 18, 2001
`\dateaustrian`	18. Jänner 2001
`\dateenglish`	18th January 2001
`\datefrench`	18 janvier 2001
`\dategerman`	18. Januar 2001

2.26.6 *musixdia

These file is automatically loaded, if you use `musixper.tex`. Here is the list of notes it provides:

- The ◆ symbol is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`y`" (think of "dyamond"). Available are `\yqu`, `\yqup`, `\yqupp`, `\yql`, `\yqlp`, `\yqlpp`, `\yzq`, `\yzqp`, `\yzqpp`, `\yqb`, `\ycu`, `\yccu`, `\ycccu`, `\yccccu`, `\ycl`, `\yccl`, `\ycccl`, `\yccccl`, `\ycup`, `\ycupp`, `\yclp`, `\yclpp`. The symbol without a stem can be obtained by saying `\ynq` and the non-spacing variant with `\zynq`.

- The ◇ symbol is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`d`" (think of *diamond*). Available are `\dqu`, `\dqup`, `\dqupp`, `\dql`, `\dqlp`, `\dqlpp`, `\dzq`, `\dzqp`, `\dzqpp`, `\dqb`, `\dcu`, `\dccu`, `\dcccu`, `\dccccu`, `\dcl`, `\dccl`, `\dcccl`, `\dccccl`, `\dcup`, `\dcupp`, `\dclp`, `\dclpp`. The symbol without a stem can be obtained by saying `\dnq` and the non-spacing variant with `\dznq`.

2.26.7 *musixeng

This package is made for music typesetters who are allergic to rest names taken from French, German or Italian[38]. It does not provide new features, only new command names.

[38] By the way, note that a *whole rest* is called *pause* in French, *ganze Pause* in German, and *pausa* in Italian.

original	*alternative*
`\PAUSe`	`\Qwr`
`\PAuse`	`\Dwr`
`\liftpause`	`\liftwr`
`\pausep`	`\wrp`
`\pause`	`\wr`
`\lifthpause`	`\lifthr`
`\hpausep`	`\hrp`
`\hpause`	`\hr`
`\qp`	`\qr`
`\ds`	`\er`
`\qs`	`\eer`
`\hs`	`\eeer`
`\qqs`	`\eeeer`

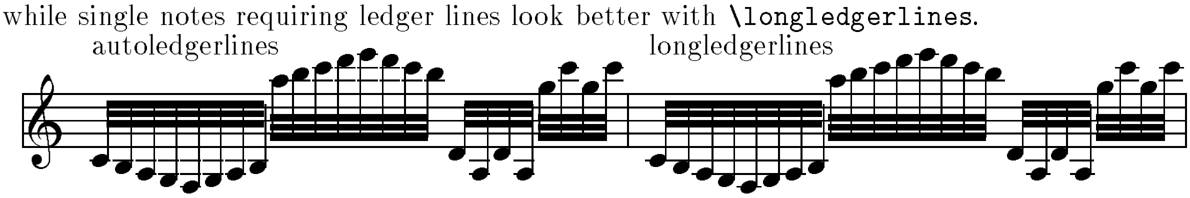
2.26.8 musixext

`\slide`{pitch}{elements}{slope[-8 to 8]} : glissando.

`\raggedstoppiece` : provides a ragged last line.

2.26.9 musixfll

`musixfll.tex` may be used to modify the automatic generation of ledger lines. Ledger lines — German Hilfslinien — are generated for notes placed above or below normal staff lines. Ledger lines normally exceed the note head by a quarter of the width of the note head. If notes are set very narrowly, the ledger lines of consecutive notes may meet. In this case a player may misread the ledger lines as normal lines. Therefore MusiXTEX shortens the ledger lines if notes are set so narrowly that the ledger lines may meet. Because MusiXTEX does not know whether consecutive notes need ledger lines, this automatic shortening may be superfluous. So this automatism may be switched off and on. After including `musixfll.tex` the automatic shortening of ledger lines is switched off. Afterwards it may be switched on again using `\autoledgerlines` and switched off again using `\longledgerlines`. Both macros have global effect.

The following example shows that narrowly set scales look better with `\autoledgerlines`, while single notes requiring ledger lines look better with `\longledgerlines`.

2.26.10 *musixgre

Provided that four line staffs (see 2.20.4) are used, *gregorian music* is frequently quoted using specific *neumes*, established in the thirteenth century, and this way of coding has been commonly used for the coding of liturgical chant in the Catholic Church until the middle of the twentieth century. Nevertheless, gregorian chant coding is not really an antique feature, since it has been codified in 1905 with recent quasi-official updates in the eighties.

REMARK: strictly writing, including gregorian chant coding in MusiXTEX is a pure nonsense. Of course, many people find it interesting to have the possibility of coding gregorian neumes together with modern music, but having a score with both a gregorian chant and a choir or *instruments is definitely heretic. The reason is that grogorian chant must be sung a cappella, without instrument and without underlying polyphony. As a matter of fact, serious music schools ceased to teach how to accompany gregorian chant approximately 20 years ago.*

However, there are still some pseudo-gregorian chants in the Catholic repertoire which deserve organ or orchestra accompaniment: these are latine speaking gregorian imitations composed in the eighteenth century, such as the famous Angel's mass of the Messe royale de Du Mont which were printed using gregorian neumes by official catholic publishers instead of using the music denotations fit for renaissance and baroque pieces...

Symbols available in MusiXTEX are mainly those of the 1905 standard, as provided by Beda Szukics:

2.26.10.1 The clefs

- The gregorian C clef: = `\gregorianCclef`, normally activated for instrument n with the command:

  `\setaltoclefsymbol` n `\gregorianCclef`

- The gregorian F clef: = `\gregorianFclef`, normally activated with the `\setbassclefsymbol` command.

2.26.10.2 The basic elementary symbols

- Diamond shaped *punctum* (of different shape compared to the percussion diamond): = `\diapunc` p .
- Square *punctum*: = `\squ` p or `\punctum` p .
- Left stemmed *virga* (not in the 1905 gregorian standard): = `\lsqu` p .
- Right stemmed *virga*: = `\rsqu` p or `\virga` p .
- *Apostropha*: = `\apostropha` p .
- *Oriscus*: = `\oriscus` p .
- *Quilisma*: = `\quilisma` p .
- *Punctum auctum* (up): = `\punctumauctup` p .
- *Punctum auctum* (down): = `\punctumauctdown` p .
- Diamond shaped *punctum auctum* (down): = `\diapunctumauctdown` p .
- *Punctum deminutum*: = `\punctumdeminutum` p .
- *Apostropha aucta*: = `\apostropha aucta` p .

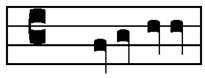
All non-*liquescens* symbols have a non-spacing variant, which are `\zdiapunc`, `\zsqu`, `\zlsqu`, `\zrsqu`, `\zapostropha` and `\zoriscus`.

2.26.10.3 The plain complex neumes

Other *neumes* can be obtained by combining two or more of these symbols. Since *neumes* have a special note head width, an additional shifting macro is provided, namely `\groff`. It is similar to `\roff`, but the offset is smaller; to implement also triple symbol neumes, another shifting macro is provided, namely `\dgroff` which performs an offset twice the offset of `\groff`..

Since most of these symbols depend on relative pitches of their components, one could not provide all compact combinations as single symbols. Following compound symbols are:

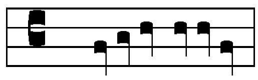
`\bivirga` n p , for example:



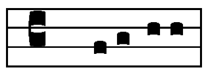
This example was coded as:

```
\instrumentnumber 1
\setstaffs 1 1
\setlines 1 4
\setclef 1{3000}
\setaltoclefsymbol 1 \gregorianCclef
\startextract
\notes \bivirga ab\enotes
\notes \bivirga cc\enotes
\endextract
```

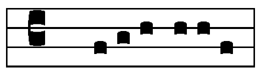
`\trivirga` n p q , for example:



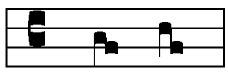
`\bistropha` n p , for example:



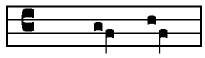
`\tristropha` n p q , for example:



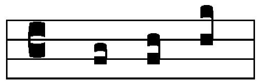
`\clivis` n p , for example:



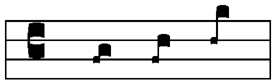
`\lclivis` n p , for example:



`\podatus` n p , for example:

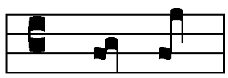


`\podatusinitiodebilis` n p , for example:

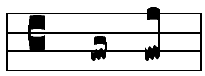


`\lpodatus` n p , for example:

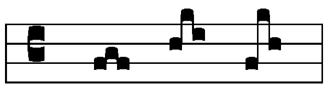
`\pesquassus` n p , for example:



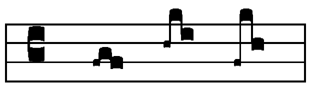
`\quilismapes` n p , for example:



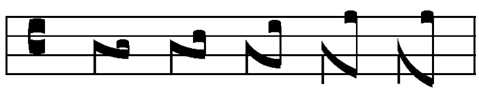
`\torculus` n p q , for example:



`\torculusinitiodebilis` n p q , for example:



`\Porrectus` n p q , for example:



coded:

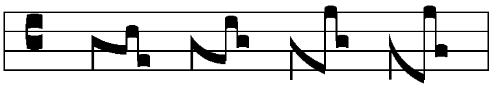
```
\notes \Porrectus bab\enotes
\notes \Porrectus bac\enotes
\notes \Porrectus bNd\enotes
\notes \Porrectus bMe\enotes
\notes \Porrectus bLe\enotes
```

`\Porrectus` exists in four different shapes, depending on the difference between first and second argument. The constraint is that

$$n - 4 \leq p \leq n - 1$$

otherwise a diagnostic occurs. Note also that `\bporrectus` provides the first curved part of the `porrectus` command, if you need it... It has two arguments, the starting pitch, the lower pitch.

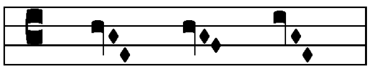
`\Porrectusflexus` $n\ p\ q\ r$, for example:



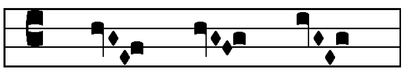
coded:

```
\notes \Porrectusflexus  bacN\enotes
\notes \Porrectusflexus  bNdb\enotes
\notes \Porrectusflexus  bMeb\enotes
\notes \Porrectusflexus  bLea\enotes
```

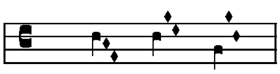
`\climacus` $n\ p\ q$, for example:



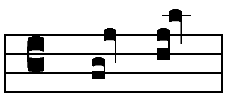
`\climacusresupinus` $n\ p\ q\ r$, for example:



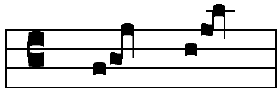
`\lclimacus` $n\ p\ q$, for example:



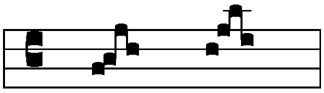
`\scandicus` $n\ p\ q$, for example:



`\salicus` $n\ p\ q$, for example:



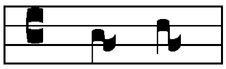
`\salicusflexus` $n\ p\ q\ r$, for example:



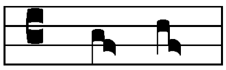
`\trigonus` $n\ p\ q$, for example[39]:

2.26.10.4 The liquescens complex neumes

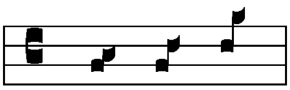
`\clivisauctup` $n\ p$, for example:



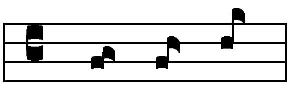
`\clivisauctdown` $n\ p$, for example:



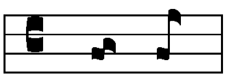
`\podatusauctup` $n\ p$, for example:



`\podatusauctdown` $n\ p$, for example:

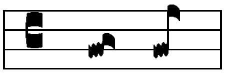


`\pesquassusauctdown` $n\ p$, for example:

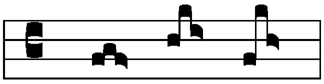


[39]The second example is in principle irrelevant, but it shows the possibilities, in case of.

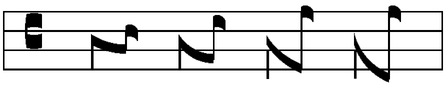
`\quilismapesauctdown` *n p* , for example:



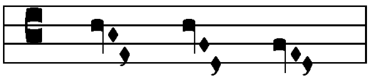
`\torculusauctdown` *n p q* , for example:



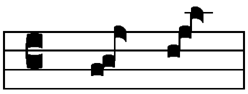
`\Porrectusauctdown` *n p q* , for example:



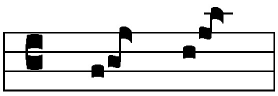
`\climacusauctdown` *n p q* , for example:



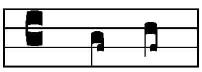
`\scandicusauctdown` *n p q* , for example:



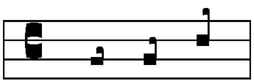
`\salicusauctdown` *n p q* , for example:



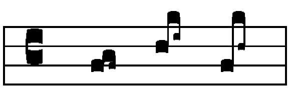
`\clivisdeminut` *n p* , for example:



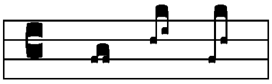
`\podatusdeminut` *n p* , for example:



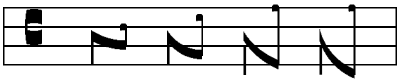
`\torculusdeminut` *n p q* , for example:



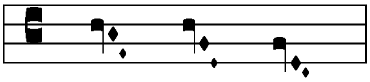
`\torculusdebilis` *n p q* , for example:



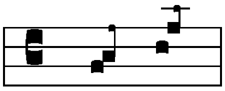
`\Porrectusdeminut` *n p q* , for example:



`\climacusdeminut` *n p q* , for example:

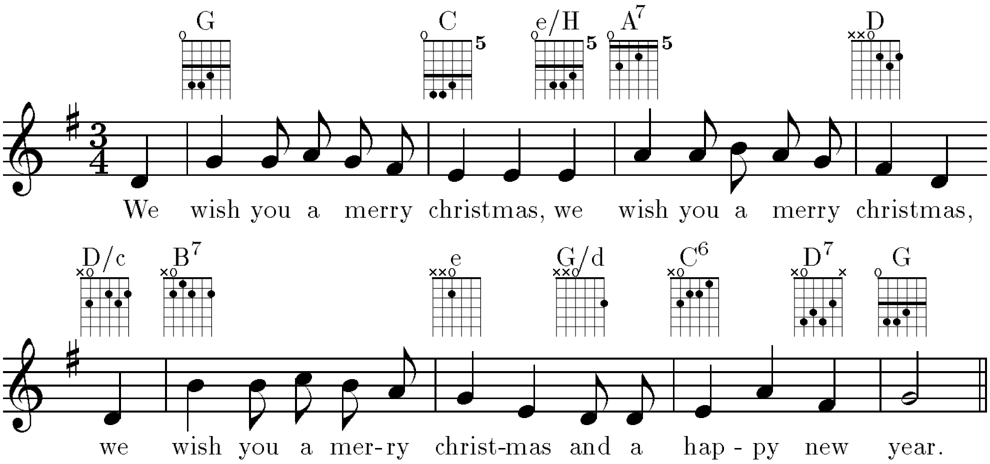


`\scandicusdeminut` *n p q* , for example:



2.26.11 musixgui

`musixgui.tex` provides some macros for typesetting *guitar tablatures*. Most times they are used above modern music. To give an example:



`\guitar` sets the grid, the chord name, the relative barre and play indicators. As example the first chord in upper example was coded as:

```
\guitar G{}o-----\gbarre3\gdot25\gdot35\gdot44
```

where the first argument is put above the grid to indicate the chord name, the second is empty (relative barre) and the other six indicates if the string is played with either `x`, `o` or `-`. The dots are set with `\gdot` *sb* where the *s* is the string and *b* is the barre. The rule is set with `\gbarre` *b* where *b* indicates the barre.

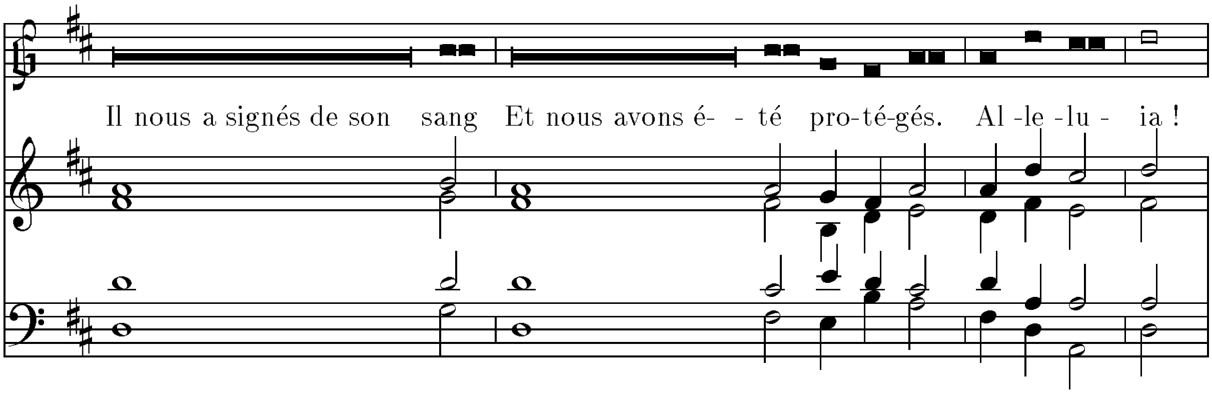
The whole chord may be vertically shifted with `\raiseguitar{`*n*`}`, where *n* is a number in units of `\internote`. It might be useful to reserve additional space above the chord by advancing `\stafftopmarg` to something like `stafftopmarg=10\Interligne`.

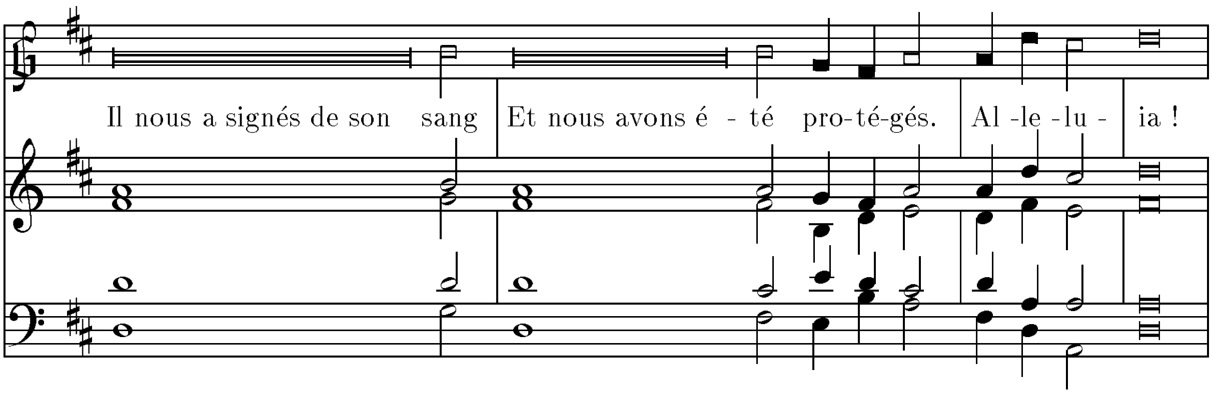
If you need the chords more often, it might be useful to define your own macros, e.g. saying:

```
\def\Dmajor{\guitar D{}x-----\gdot42\gdot53\gdot62}%
```

2.26.12 *musixlit

Provides a way of coding intermediate between gregorian and baroque/romantic, still used[40] for liturgical works!





This package provides:

- `\oldGclef` which replaces the ordinary G clef with an old one, using (for instrument 2 as an example): `\settrebleclefsymbol2\oldGclef`
- `\cqu` *p* provides a square headed quarter note with stem up at pitch *p*.
- `\cql` *p* provides a square headed quarter note with stem down at pitch *p*.
- `\chu` *p* provides a square headed half note with stem up at pitch *p*.
- `\chl` *p* provides a square headed half note with stem down at pitch *p*.
- `\cnqu` *p* and `\cnql` *p* provide a stemless square headed quarter note at pitch *p*.
- `\cnhu` *p* and `\cnhl` *p* provide a stemless square headed half note at pitch *p*.

[40]Was written by A. Egler. To my knowledge (D. T.), liturgical works are either written with modern music quotation, or using the gregorian *neumes* which have been normalized in 1905.

- `\Hpause` *p n* provides an arbitrary length pause at pitch *p* and of length *n* `\noteskip`. However, in the first of the above example, this feature has been used to denote an arbitrary length note rather than a rest!

- `\Hlonga` *p n* provides an arbitrary length note at pitch *p* and of length *n* `\noteskip`. This feature has been used to denote an arbitrary length note in the second of the above examples.

- `\shortbarrules` has been used to provide bar rules shorter than the staff vertical width.

- `\interbarrules` has been used to provide bars between the staffs, rather that over them. This is an arbitrary question of taste...

2.26.13 musixmad

`musixmad.tex` increases the number of instruments, slurs and beams up to twelve.

2.26.14 musixper

Provides specific percussion symbols, especially `\drumclef` (used in section 2.21.2.3). These definitions cause problems since drum and percussion notation is not standardized. So it might be clever to specifiy the names of instruments for each used symbol.

- The ⊗ symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`dc`" (think of *DiagonalCross* for e.g. closed *hihat*). Available are `\dcqu`, `\dcql`, `\dcqb`, `\dczq`, `\dccu`, `\dcccu`, `\dccl` and `\dcccl`.

- The ⊗ symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`dh`" (think of *DiagonalcrossHalfcircle* for e.g. half open *hihat*). Available are `\dhqu`, `\dhql`, `\dhqb`, `\dhzq`, `\dhcu`, `\dhccu`, `\dhcl` and `\dhccl`.

- The × symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`do`" (think of *DiagonalCross and O for circle* for open *hihat*). Available are `\doqu`, `\doql`, `\doqb`, `\dozq`, `\docu`, `\doccu`, `\docl` and `\doccl`.

- The × symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`x`" (e.g. for spoken text of songs). Available are `\xqu`, `\xql`, `\xqb`, `\xzq`, `\xcu`, `\xccu`, `\xcl` and `\xccl`.

- The ⊗ symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`ox`" (.!.). Available are `\oxqu`, `\oxql`, `\oxqb`, `\oxzq`, `\oxcu`, `\oxccu`, `\oxcl` and `\oxccl`.

- The ▰ symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`ro`" (think of *RhOmbus* for e.g. shaker). Available are `\roqu`, `\roql`, `\roqb`, `\rozq`, `\rocu`, `\roccu`, `\rocl` and `\roccl`.

- The ▵ symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`tg`" (think of *TrianGle* for e.g. rattle). Available are `\tgqu`, `\tgql`, `\tgqb`, `\tgzq`, `\tgcu`, `\tgccu`, `\tgcl` and `\tgccl`.

- The + symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceeded with a "`k`" (for bongos). Available are `\kqu`, `\kql`, `\kqb`, `\kzq`, `\kcu`, `\kccu`, `\kcl` and `\kccl`.

2.26.15 musixpoi

Adds pointed compact definitions of notes. Available are `\ccup`, `\zccup`, `\cclp`, `\zcclp`, `\ccupp`, `\zccupp`, `\cclpp`, `\zcclpp`, `\cccup`, `\zcccup`, `\ccclp`, `\zccclp`, `\cccupp`, `\zcccupp`, `\ccclpp`, `\zccclpp`, `\ccccup`, `\zccccup`, `\cccclp`, `\zcccclp`, `\ccccupp`, `\zccccupp`, `\cccclpp` and `\zcccclpp`.

2.26.16 musixstr

This file provides a set of indications for *string instruments* (e.g. violin) execution, provided by Werner Icking.

The symbols described below should be posted at the wanted place using `\zcharnote`p{$< command - name >$}.

: `\AB` **or** `\downbow` down-bow

V : `\AUF` **or** `\upbow` up-bow

: `\SP` at the top of bow

: `\FR` at the nut of bow

$\longleftrightarrow$ **or** $\leftrightarrow$: `\GB` **or** `\Gb` whole bow

$\longleftarrow$ **or** $\leftarrow$: `\UH` **or** `\Uh` lower half of bow

$\longrightarrow$ **or** $\rightarrow$: `\OH` **or** `\Oh` upper half of bow

$\vdash\!\!-\!\!\dashv$ **or** $\vdash\!\dashv$: `\MI` **or** `\Mi` middle of bow

$\longleftarrow\!\dashv$ **or** $\leftarrow\!\dashv$: `\UD` **or** `\Ud` lower third of bow

$\longmapsto$ **or** $\mapsto$: `\OD` **or** `\Od` upper third of bow

$+$: `\Pizz` left hand pizzicato or trill

2.26.17 musixsty

This file is made for non TEXperts and/or lazy score typesetters. Although related to MusiXTEX, it has little to see with music, but it helps writing all surrounding texts, like *titles*, *author names*, historical comments, etc. It provides

- a set of font definitions of common use, such as `\tenrm`, `\eightrm`, etc.,
- a reasonable setting of `\hsize`, `\vsize`, `\hoffset`, `\voffset` dimensions in order to have a good layout fit for European A4 paper[41]
- a set of text size commands:

  `\eightpoint` which sets the usual `\rm`, `\bf`, `\sl`, `\it` commands to 8 point font size;

  `\tenpoint` which sets the usual `\rm`, `\bf`, `\sl`, `\it` commands to 10 point font size;

  `\twlpoint` to get 12 point font size;

  `\frtpoint` to get 14.4 point font size;

  `\svtpoint` to get 17.28 point font size;

  `\twtypoint` to get 20.74 point font size;

  `\twfvpoint` to get 24.88 point font size;

[41] People addicted to *legal* or other paper sizes should correct it for their own purpose.

- a set of commands to make easy piece titles:
  - `\author` or `\fullauthor` to be put at the right of the first page, below the title of the piece; the calling sequence is, for example:

    `\author{Daniel TAUPIN\\organiste \'a Gif-sur-Yvette}`

    where the `\\` makes the author's name displayed on two or several lines.
  - `\shortauthor` to be put at the bottom of each page,
  - `\fulltitle` which is the big main title of the piece,
  - `\subtitle` is displayed below the main title of the piece,
  - `\shorttitle` or `\title` which is the title repeated at the bottom of each page,
  - `\othermention` which is displayed on the left of the page, in front of the author's name (it may contain several `\\` to display it on several lines,
  - `\maketitle` which displays all the previous stuff.
- Some additional commands to make *footnotes*. These commands are
  - The normal Plain-TEX `\footnote` command which has two arguments — not only one as in LATEX — namely the label of the footnote, i.e. any sequence of characters and not only figures, and the text of the footnote.

    IMPORTANT: the `\footnote` *command does not work inside boxes*[42]*, therefore this command must not be issued within music. But another alternate feature is provided (see below).*
  - The `\Footnote` command, which counts the footnotes and uses a number as the label of the foot note (equivalent to LATEX's `\footnote` command). The same restriction applies concerning footnotes within the music coding.
  - The `\vfootnote` command, taken from the Plain-TEX, which makes the footnote itself at the bottom of the current page, but does not put the footnote label at the place it is referred in the main text.

    Thus, if a footnote is needed whose reference lies inside the music itself, the music typesetter must perform it in two steps:

    1. quote the reference inside the music, using `zcharnote` for example,
    2. post the footnote itself, using `\vfootnote` outside the music, either before `\startpiece` (or `\debutmorceau` with `musixcpt.tex`) or between `\stoppiece` and `\piececont` or equivalent commands.

Note that `musixsty` should not be used with LATEX.

2.26.18 musixtri

Provides triple pointed note symbols. Available are: `\lpppt`, `\whppp`, `\zwppp`, `\huppp`, `\hlppp`, `\zhppp`, `\zhuppp`, `\zhlppp`, `\quppp`, `\qlppp`, `\zquppp`, `\zqlppp`, `\zqppp`, `\cuppp`, `\zcuppp`, `\clppp`, `\zclppp`, `\qbppp` and `\zqbppp`.

[42] This is not a TEX-bug, this is a feature!

2.27 Often reported problems and discussion faxilities

2.27.1 Newsgroups

No specific newsgroup exist concerning M$^{\mathrm{u}}$TEX, MusicTEX or MusiXTEX; nevertheless W. Icking saw articles on this topic in `comp.text.tex` as well as in `de.comp.text.tex`.

2.27.2 Mailing list

A mailing-list does exist which was formerly dedicated to M$^{\mathrm{u}}$TEX, and now to MusicTEX and mostly to MusiXTEX. You can subscribe to it by sending an e-mail to `mutex-request@gmd.de` with the message line

`subscribe`

2.27.3 FAQs

`gmd.de`, too, has a FAQ which is presently out-of-date and quite unusable since it displays many questions and comments without giving the answer. However, it will be probably enhanced in a near future. This FAQ can be accessed using `ftp`:

`ftp://ftp.gmd.de/music/mutex/faq`

and the archive of the mailing list may be accessed also by `ftp`:

`ftp://ftp.gmd.de/music/mutex/archive`

or:

`http://www.gmd.de/Mail/`

`ftp.gmd.de` also contains a lot of stuff concerning music and typesetting music, which can be accessed by WWW at:

`http://www.gmd.de/Misc/Music/`

Chapter 3

Installation

3.1 *Getting the stuff

As seen before, all the files are available at *anonymous ftp* `hprib.lps.u-psud.fr` (193.55.39.152) in the directory

`/pub/musixtex`

The `*.mf` source files are in

`/pub/musixtex/mf`

and the `*.tfm` files are in

`/pub/musixtex/tfm`

Besides, this manual is in

`/pub/musixtex/musixdoc.ps`

Another directory, namely `/pub/music_zips` normally contains `musixtex.zip` which contains all the distribution for PC (MS-DOS) computers. This is only for `ftp`-ing convenience since all source files are directly available in the general directory and subdirectories. In addition a set of examples (i.e. the files whose name does not begin with "`musix`") is packed into `musixexa.zip` and this manual is also in `musixdoc_ps.zip` (the PostScript version) and `musixdoc_lj.zip` (the PCL version for Laserjets). Finally the PK files of specific fonts are provided in `musixpk.zip`, `musixpk3.zip` (300 and 360 dpi) and `musixpk6.zip` (600 dpi); getting these files is useless if you are able to METAFONT the files whose `*.mf` are provided in the main package.

CAUTION: the zipped files are provided in the PC line coding, i.e. with `<cr><lf>` at line ends, rather than single `<lf>` as in the UNIX coding. This means that UNIX users must carefully unzip these files using the "`-a`" (our "`-aa`" options of `unzip` to get correct coding of the source files. Other wise they would find a lot of "`^M`" in their files, which may cause further troubles.

Besides, /unix users can also — when in directory `/pub` — ask for:

`get musixtex.tar` or

`get musixtex.tar.Z` or

`get musixtex.tar.gz` and even

`get musixtex.zip`

Then, the `hprib.lps.u-psud.fr` server will perform required collection and/or compression "on-the-fly". Note that in that case the source files will be in the UNIX coding rather than PC coding of end of lines. Usually, this is harmless for PC editors and TeX motors, but who knows...

Fonts are provided in `musixtex.zip` as `*.mf` files but also as `*.tfm` and `*.pk` files for 300 dpi printers or previewers. Other values of the dpi parameter are also provided in `musixpk*.zip`.

Needed fonts are `musix20`, `musix16`, `musix13`, `musix11`, `musix24`, `musix29`, `musixspx`[1], `xslhu20`, `xslhu16`, `xslhu24`, `xslhu29`, `xslhd20`, `xslhd16`, `xslhd24`, `xslhd29`, `xslu20`, `xslu16`, `xslu24`, `xslu29`, `xsld20`, `xsld16`, `xsld24`, `xsld29`, `xslz20` and `xslhz20`.

3.2 Installing the fonts

All files with the extension `.tfm`[2] must be copied in that directory, in which the other `.tfm`'s are.

If you get the error message:

```
! Font ... not loadable: Metric (TFM) file not found.
```

this means you have not succeeded to install the `.tfm`'s in the right directory. This is not a MusiXTEX error, but a local TEX implementation error/misunderstanding. Then read or re-read your TEX installation doc, put the `tfm`'s at the very right place and start again.

The `.tfm` only contains the width, height and depth of every character of a font and is the only needed font-file for TEXing. For previewing and/or printing you need the pixel fonts. On most systems they are packed and have the extension `.pk`(but perhaps you need the `.gf`-files or `.300`-files or other: Ask your local TEX-wizard[3]!) Most needed and spreaded are fonts for `dvi`-driver with resolution of 300dpi. Using emTEX associated drivers the `.pk`-fonts have to be copied in `...\pixel.lj\300dpi\` as a default, and in general in a directory specified by the environment variable `DVIDRVFONTS`[4]. For other resolution refer your local manual or TEX-wizard.

3.3 Building a format

Introducing these files in a *format* (with INITEX) is a mean of saving computer time and memory. Besides, if you include `musixcpt.tex` in your format, you will have a format compatible with MusicTEX and — provided you made the symmetrical format for MusicTEX — you can compile exactly the same source files with both MusicTEX and MusiXTEX, which is a good means of finding whether some strange behaviour is specific to one implementation of the other, or whether you made some general mistake[5]...

However, although having a MusiXTEX format saves much time, it can also lead to time wasting weird things in case of updates : if you update your source of `musixtex.tex` but not the format `musixtex.fmt`, then running the command `musixtex` will invoke the old unmodified format...

3.3.1 Starting from nil

1. Build up a file called `musixtex.ini` with following content:

[1]Inside this file are two types of piano brackets available. The default you can see printing this doc, the other you can see printing `musicdoc`. If you prefer the other, which are designed by Stanislav Kneifl, follow the instruction inside `musixsps.mf`

[2]TEX font metric, needed directly from binary for TEXing the examples or the doc.

[3]I have read a mail from somebody, who has said, that he is able — because he is a professional — to write an `install` for installation of packages like MusiXTEX. Really? Do it.

[4]Look at `dvidrv.doc` for more information.

[5]Not all authors have the same opinion :-). But I was told, that my opinion doesn't count, so forget it – A.E.

```
\input plain % or dcplain or eplain or see later a safer way of doing
\input musixtex
% here can you add your most needed files from extension lib
% musixcpt at very last !

\def\fmtname{musixtex}\def\fmtversion{T.101} \dump
```

2. `initex musixtex.ini`[6].

3. `tex &musixtex` *jobname*[7].

[6] depends on your implementation
[7] depends on your implementation

3.3.2 Starting from your usual plain format

1. First, try to find — on your favourite system — whether `tex` is an executable routine, or a `tex.bat` command in MS-DOS or a *shell* procedure under UNIX.

2. If `tex` is a command try to find the "`initex`" local command: usually it is either `initex` or `tex -i`.

3. Try to find the name of the "plain TEX" format (usually posted when TEXing anything).

4. Then, *mutatis mutandis*, assuming the "`initex`" command has the name `initex` and the "plain TEX" format is `plain`, run the shell command:

   ```
   initex  \&plain musixtex.ins
   ```

   which will produce a format file `musixtex.fmt` which you shall put in the same directory as the others formats (hoping you have the access rights...). Note — in UNIX systems — the backslash before the `&` which tells the system to consider this character as a member of the command, not a batch execution indication. Once this is done, you can MusiXTEX any score you have written using a command such as:

   ```
   tex \&musixtex my-score.tex
   ```

   that is, specifying your new format `musixtex.fmt` instead of the usual `plain.fmt`.

   For MS-DOS/`emTeX` users the format building command is:

   ```
   tex386 -i &plain musixtex.ins
   ```

   (you can change `plain` into `dc-plain` or any other plain-like format you have) then:

   ```
   copy musixtex.fmt \emtex\btexfmts\*.*
   ```

   and the `musixtex.bat` command can be

   ```
   if exist %1.tex goto tex
   goto end

   :tex
   if exist %1.mx1 del %1.mx1
   if exist %1.mx2 del %1.mx2
   tex386 -mt20000 &musixtex %1 %2 %3 %4 %5 %6 %7 %8 %9

   if errorlevel 2 goto end
   musixflx %1

   if errorlevel 1 goto end
   tex386 -mt20000 &musixtex %1 %2 %3 %4 %5 %6 %7 %8 %9

   :end
   ```

Don't expect a perfect package or miracles.
There will be enough unexpected miracles.
EBERHARD MATTES

Chapter 4

Examples

Due to compatibility problems with LaTeX (used to produce this notice) large examples must be TeX-ed separately, i.e. using plain TeX and not LaTeX. Therefore, the MusiXTeX future user is suggested to produce some of the following examples and to look carefully at the way some special features have been coded.

When producing this examples, care should be taken about the fact that several given files are supposed to be included (by means of `\input`) in other files. Thus the only good files to be directly TeX-ed are those which begin with `\input musixtex` or `% \input musixtex`. This latter command is often commented out so that the examples can be run either using a MusiXTeX format including `musixcpt.tex` and `musixsty.tex` — namely the format generated by `musixtex.ins` — or using a MusicTeX format including `musictrp.tex`, `musicvbm.tex` and `musicsty.tex` — namely the format generated by `musixtex.ins`.

In addition, it must be noted that most DVI previewers and laser printers have their origin at one inch below and one inch right of the right upper corner of the paper, while the musical examples have their upper left significant corner only at one centimeter right and below the left top of the paper. Therefore, special parameters have to be given to the DVI transcription programs unless special `\hoffset` and `\voffset` TeX commands are introduced within the source TeX text.

4.1 Clean MusiXTeX examples

- `avemaria.tex` to get the "Méditation" (alias "Ave Maria") by Charles Gounod for organ and violin or song in clean MusiXTeX-input.

- `traeumer.tex` to get the famous "Träumerei" by Robert Schumann for piano, in genuine MusiXTeX-input but with some additions to perform ascending *crescendos*.

- `parnasum.tex` to get the first page of "Doctor gradus ad Parnassum" by Claude Debussy for piano.

- `glorias.tex` to get a local melody for the French version of *Gloria in excelsis Deo*, showing use of lyrics commands. `gloriab.tex` is the same, with organ accompaniment.

- Other clean MusiXTeX examples seem to be no more compatible with the present version, but they should be updated in a near future.

These examples are packed (zipped) in the `musixexa.zip` file of the MusiXTeX distribution.

4.2 Nearly compatible MusicTEX examples

These are examples using not only `musixtex` but also `musixcpt` and `musixsty`, which means their coding is easily understandable for MusicTEX previous users. However, they cannot be run as is with MusicTEX, since a few specific commands have been included, either for spacing, or for oblique slurs. They are given in `musixexa.zip`, for example:

- `gymnoman` : an imitation of Erik SATIE's *Gymnopédies*,
- `canticor.tex/canticox.tex` : a famous tune by Georg Friedrich HAENDEL,
- `ilfaitda.tex/ilfaitdx.tex` : a famous choral by J.-S. BACH adapted with French lyrics for French churches,
- `widor_20.tex` : the famous toccata for organ by Charles-Marie WIDOR in 10 pages (more compact than commercial editions, and easier to play if nobody can turn the pages), and `widor_16.tex` which is the same in 16 pt size, i.e. more compact.
- `souveni*.tex` : a nice French tune of the fifties, by Oreste ROSSI and Fernand CAPITANI with several transpositions.
- `marcello.tex` : the adagio of Benedetto MARCELLO's oboe concerto, transcribed for organ.
- `hymnepas.tex` : a modern Eastern canticle by A. GOUZES and PALESTRINA, showing heavy use of `\hardlyrics` for choral music.

4.3 Compatible MusicTEX examples

They are all the examples provided in the MusicTEX distribution. To run them the easiest way is to have both a `musictex` and a `musixtex` format as described in 3.3. Otherwise, insert

```
\input musixtex
\input musixcpt
\input musixsty
```

at the beginning of each example, and TEX it.

4.4 Looking at examples before installing MusiXTEX

Most of the above examples (and some others) are posted, ready to print, as PostScript files in:

- `ftp://hprib.lps.u-psud.fr/pub/music_scores.zip/*/*/*.zip` (zipped PostScript 300dpi files)
- `ftp://hprib.lps.u-psud.fr/pub/music_scores.gz/*/*/*.ps.gz` (gzipped PostScript 300dpi files)

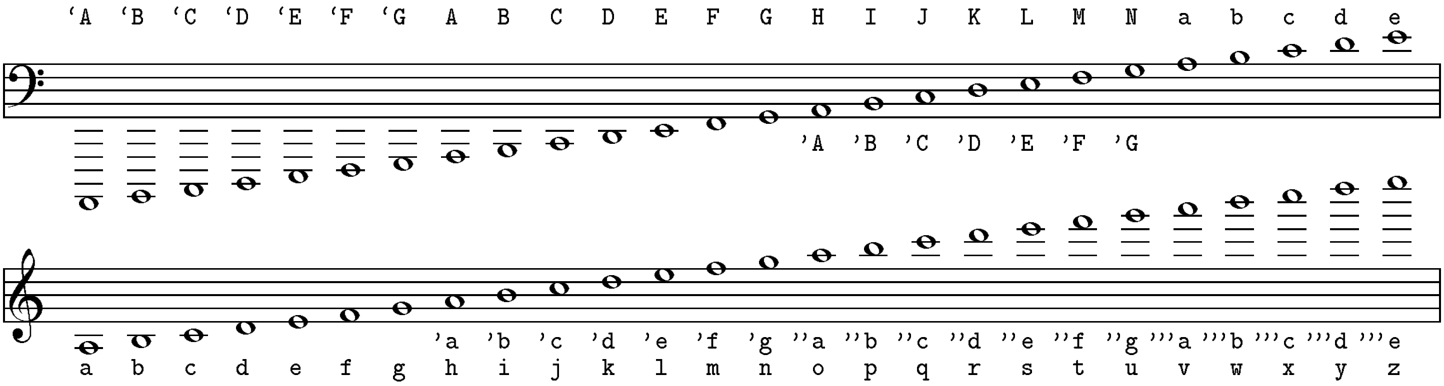
Printing them on any PostScript printer (needs A4 paper) is a good means of getting an overview of MusiXTEX's capabilities.

Chapter 5

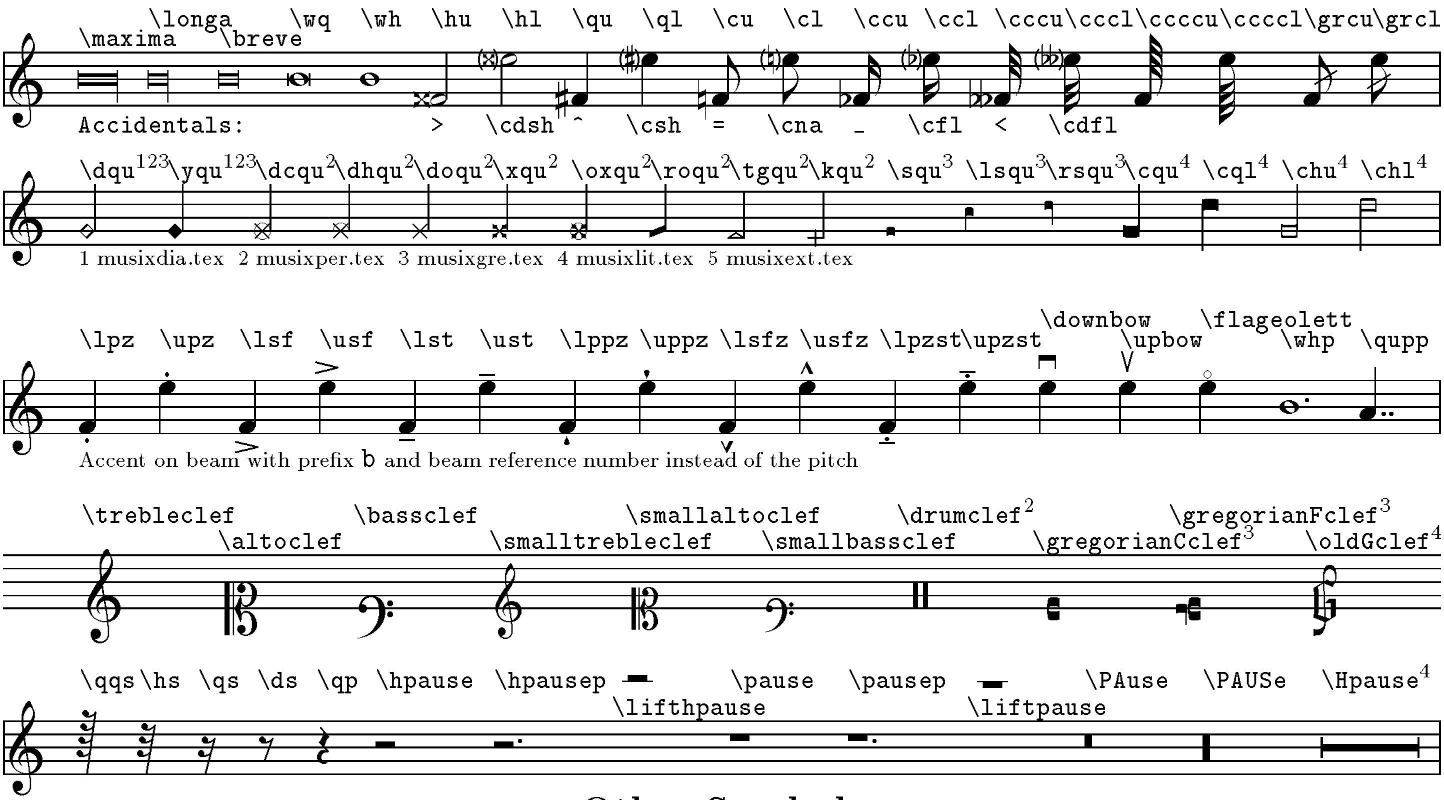
Index

Chapter 6

Pitches



Notes, Accidentals, Accents, Clefs and Rests



Other Symbols