A new font selection scheme for TeX macro packages — the basic macros

Frank Mittelbach
Rainer Schöpf
Johannes Gutenberg Universität Mainz

Abstract
We have implemented a new font selection scheme for TeX and its macro packages. This scheme allows font family, series, shape, and size to be specified independently. Additionally, it is not necessary to preload all math fonts.

Contents

1 Introduction 222
2 The User Interface 222
3 Setting up a new format 223
4 Concept of the implementation 225
5 Preliminary macros 227
6 Macros for setting up the tables 228
7 Selecting a new font 229
8 Assigning math fonts to versions 234

1 Introduction
In traditional typesetting one distinguishes four parameters to describe a font: the font family (e.g. computer modern), the font series (e.g. roman or sansserif), the font shape (e.g. normal or bold), and the font size. This distinction is not always unique: take for example the slanted typeface \textsf{LATeX} uses. This can be seen as the sloped shape of series roman or as the normal shape of series sloped.

Recently several people have asked how to use such a scheme in \textsf{LATeX}. Unfortunately the current implementation of \textsf{LATeX}'s font selection scheme does not allow incorporation of this concept.

When typesetting math formulas, one usually needs many more fonts than for ordinary text. In the TeXbook Donald Knuth says:

All characters that are typeset in math mode belong to one of sixteen families of fonts\footnote{Emphasis by DEK}, numbered internally from 0 to 15.

The use of the word family in this context is unfortunate; it conflicts with the font families we are talking about. To avoid confusion we will always speak of font families from the typesetter's point of view.

For math we speak of math groups each connected to three fonts called the \texttt{textfont}, the \texttt{scriptfont}, and the \texttt{scriptsiprofont}. From the user's point of view, math formulas consist of characters coming from specific math alphabets (e.g. those selected by \texttt{\cal}) and of symbols (e.g. \texttt{\sum}) selected by a special control sequence and scattered over a number of fonts.

All fonts that can be used together in one math formula form a version. Versions can only be switched outside math formulas. Standard \textsf{LATeX} provides two versions: normal and bold.

2 The User Interface
The commands described in the next subsections are primitives used to build up more powerful interfaces. But they are all user accessible. We used these commands to construct two interfaces for \textsf{LATeX}: one is mimicking the old font selection (e.g. \texttt{\bf} is used to switch to the font \texttt{cmbr..}), the other one implements an orthogonal font selection scheme (here
\bf means: change the current shape and select a new font but leave family, series and size untouched.) Details can be found in the article “The new font family selection — User Interface to standard \TeX".

2.1 Selection of a new font

Selecting a new font is done in two independent steps. First you have to change the values for family, series, shape and/or size and then execute a macro which uses the new values to select the desired font. If you don’t use this macro the font will not be changed.

The first step is done with the macros \family, \series, \shape and \size. For example, if you want to switch to the 'sansserif' series you have to say \series(sansserif). Except for \size, all those macros have one argument, namely, the desired family, series or shape, respectively. The macro \size is somewhat special because we decided that it would be better to force the user to specify a new size and a \baselineskip together for this size, so the macro has two arguments.

All four macros will silently accept their arguments. Warning messages are generated in the second step when the actual font selection is carried out.

To select a (new) font one has to call the \selectfont macro. This macro looks up the current family, series, shape and size, possibly changed by one of the above mentioned commands, and switches to this font, provided the selected combination of family, series and shape is known to the system. If it is unknown, a warning will be printed, and up to three new trials are made to find a substitute. This is done by changing to \default@shape, then to \default@series and as a last resort to \default@family. At least this combination must have been defined, otherwise we will find ourselves in an endless loop.2

It may still be, however, that the size requested is not specified in the table. This will lead to an error, and the font given by \default@errorfont will be selected. All four defaults are given private names (names containing an @) to emphasize that their values should be changed only by the “local wizards”.

The selection scheme described above may seem unnecessarily complicated. But consider the following example: you are now reading a sentence typeset in the cmr10 font, that is family ‘computer modern’, series ‘roman’, shape ‘normal’ and size ‘10’. If we want to switch to ‘typewriter italic’ we say \series(typewriter) \shape(italic) and then \selectfont. To avoid the call to \selectfont we would have to embed it in the definition of \series, etc. But this means that either cmr10 or cmr10 would be unnecessarily selected (and probably loaded).

As mentioned before, these commands are primitive; they should be used to define higher level commands for a special application. For example \TeX’s \bf command can be defined as

\begin{verbatim}
def\bf{\series{roman}\
\shape{boldtext}\
\selectfont}
\end{verbatim}

to work in the same way as before in \TeX.3 As an alternative, the definition might be

\begin{verbatim}
def\bf{\shape{boldtext}\
\selectfont}
\end{verbatim}

which will change to the ‘bold extended’ shape in the current family, series and size.

2.2 Changing the math version

\mathversion switches to another math version, e.g.,

\begin{verbatim}
\mathversion{bold}
\end{verbatim}

will switch to version ‘bold’ provided that it is defined. This command can be used only outside of math formulas. As an example we give the definitions of Standard \TeX’s \boldmath and \unboldmath macros in terms of \mathversion. For this we must assume that two versions ‘cmnormal’ and ‘cmbold’ are already defined.4

\begin{verbatim}
def\boldmath{\nomath\boldmath\
\mathversion{cmbold}}
def\unboldmath{\nomath\unboldmath\
\mathversion{cmnormal}}
\end{verbatim}

3 Setting up a new format

3.1 Defining a new family/series/shape combination

Assume that you want to define the combination family ‘computer modern’, series ‘concrete’, shape ‘italic’. In the present case we have to write

\begin{verbatim}
\new@fontshape{cm}{concrete}{italic}\
<S>1ccr6%
\end{verbatim}

This can be fixed easily but we are not sure if it’s worth the effort. The defaults shouldn’t be changed by an ordinary user job, and it’s not necessary to provide code to check a format file.

Actually this definition behaves differently when used in math mode, because then no bold face is selected. We will see a correct definition later.

4 The \nomath command used here issues a warning if these commands are used in math mode.
The general form of the specification in the fourth argument of `\newfontshape` is

\[
<size>(external font name)
\]

You are totally free in what you write between the `<>` to denote the size.

If you look closely at the example given above you'll notice that the first three lines (for sizes 5, 6, and 7) seem to be wrong: they start with a 1 and the external font names are incorrect. This is a special feature of the font selection code that allows font substitution. The numbers in front of the external font name mean:

0 No effect. Same as no number at all.
1 Issue a warning that the requested family/series/shape combination is not available in this size and use the font given instead.
2 Issue a warning that the requested family/series does not contain the requested shape and use the font given instead.

Additionally, for every family/series combination there exists a so-called 'extra' macro that is used to set parameters, etc. common to all shapes and sizes, e.g. inhibiting hyphenation for typewriter fonts. Its argument is the internal font name.

\[\text{\texttt{\textbackslash extra@def \{cm\}\{\texttt{\textbackslash typewriter}\}\{\texttt{\textbackslash hyphenchar#1\textbackslash m@one}\}}}\]

### 3.2 Preloading fonts

The macro `\preload@sizes` provides an easy way to specify fonts that should be preloaded when dumping a format file. It is used as follows:

\[
\text{\texttt{\textbackslash preload@sizes\{\{family\}\{\{series\}\{\{shape\}\{\{list of sizes\}\}}}}}
\]

where the elements of `\{list of sizes\}` are delimited by commas. Note that it makes no difference for your documents whether you preload a font or load it on demand. In the latter case, however, processing documents takes more time.

### 3.3 Defining math groups

To specify fonts for math, other primitive commands are provided. They all have @ characters in their names; i.e. they will not normally be accessible to the user, but will be when making format files or style files.

Math fonts can be divided in two classes: fonts that are accessed via `\mathchardef` and those that are selected only via a `(math alphabet identifier)`.

As we already mentioned, all math fonts come in groups of `\textfont`, `\scriptfont`, and `\scriptscriptfont`. A new math group is defined by the command

\[
\text{\texttt{\textbackslash newmathgroup\{\{math group number\}\{\{math group number\}}} is a control sequence that is assigned a number that from now on will denote this group. It is also possible to use an explicit number, i.e. a sequence of digits, instead of this control sequence to stand for this group. However, the first alternative is generally superior since `\newmathgroup` always assigns a previously unused number to this control sequence. The second alternative is normally used for groups 0, 1, 2, and 3 which have a special meaning to \TeX.

To specify the fonts of this group, the commands `\define@mathgroup` (for the first class) and `\define@mathalphabet` (for the second class) are available.

Take for example one of the cmsy.. fonts, i.e. the standard math symbol fonts in the computer modern family. (They also contain the calligraphic alphabet.) This font must be loaded prior to its use because of the `\mathchardef` commands in `plain.tex`. To achieve this we write

\[
\text{\texttt{\define@mathgroup \{\texttt{\{cmnormal\}\{2\}}}\{\texttt{\{cm\}\{\texttt{\textbackslash mathsymbol\}\{\texttt{\textbackslash normal\}}}\}}}
\]

This can be read as: define the group number 2 in the 'cmnormal' version to consist of fonts with family 'cm', series 'mathsymbol', and shape normal. The actual sizes for `\textfont`, `\scriptfont`, and `\scriptscriptfont` will be determined when the group is selected.

If you want to access such a math group also via a `(math alphabet identifier)` you must define this control sequence to switch to the corresponding internal group `(number)`, viz.

\[
\text{\texttt{\define@mathgroup \{\texttt{\{math alphabet identifier\}\{\{number\}\}}}\}}
\]

Returning to our example: to define \ Cal to select the calligraphic alphabet (A, B) in a formula one has to add the definition

\[
\text{\texttt{\define@mathgroup \{\texttt{\textbackslash cal\{\texttt{\{group\texttt{\textbackslash 2}\}}\}}}\}}
\]

If we want to declare a group that is accessed always by a `(math alphabet identifier)` the `\define@mathalphabet` macro should be used. Since the corresponding fonts are not accessed
by \texttt{\textbackslash mathchardef} commands, there is no need to preload them. Loading can be done by the \emph{(math alphabet identifier)}. The macro \texttt{\textbackslash define@mathalphabet} is similar to \texttt{\textbackslash define@mathgroup}. If one uses a macro \texttt{\textbackslash sfmath} to select sansserif letters in a formula one has to make a declaration like
\begin{verbatim}
define@mathalphabet{cmnormal}\sfmath
\end{verbatim}
Here \texttt{\sfmath} is the new \emph{(math alphabet identifier)}. The \emph{(math group number)} that must previously be defined using \texttt{\textbackslash new@mathgroup}.

The text size in math formulas is always determined to be the \emph{size} of the text outside. The sizes for subscripts, etc., i.e. the script and the scriptscript size, must be specified additionally. The macro \texttt{\define@mathsizes} is made for this purpose. It takes three arguments: a text size, the corresponding script size and scriptscript size, e.g.
\begin{verbatim}
define@mathsizes(10){7}{5}
\end{verbatim}
defines the script and scriptscript sizes for a text set in \emph{size} '10' to be '7' and '5', resp.

When a \emph{size} change occurs, not only the current font must be switched but also all math fonts which can be selected via special symbols. On the other hand, it may be that math fonts are only available or only used in certain \emph{sizes}. For the other \emph{sizes} we do not need to switch the whole set of math fonts.\footnote{Think of a special \emph{size} used only in titles with no formulas at all.} To specify this we provide the command \texttt{\define@nomathsize} that takes only the text size and inhibits math font switching for this \emph{size}.

If there is more than one \emph{version} provided then you better define all \emph{groups} for every \emph{version}. Otherwise switching the \emph{version} (by using \texttt{\textbackslash mathversion}) will not reset these \emph{groups} properly. E.g., in the \LaTeX{} implementation we therefore have a line
\begin{verbatim}
define@mathgroup(cm@bold)2%
\end{verbatim}
It goes without saying that the \emph{family/series/shape} combination must have been defined previously by a \texttt{\textbackslash new@fontshape} command.

\section{Concept of the implementation}
\subsection{Handling the font tables}
The first problem we had to solve was how to handle such a huge number of fonts. To implement the four dimensional grid of fonts we maintain an association list\footnote{Lisp hackers note!} (i.e. a list of pairs) with elements (size, external font name) for every combination of font \emph{family/series/shape}. We do not redefine the font changing commands: these commands select the correct font by looking into the association list corresponding to the current font \emph{family/series/shape} combination. This association list is hidden in a macro. Its precise form is as follows: For every \emph{(size)} we have a string of the form
\begin{verbatim}
\langle(size)\rangle\langle(external font name)\rangle
\end{verbatim}
This strings are simply concatenated to form one long string of characters. In this way all necessary information is available. But this solution would take up far too much of \TeX{}'s valuable main memory. Therefore we use a trick, the same trick that is used in plain \TeX{}'s \texttt{\newhelp} macro: we enclose the list of characters by \texttt{\csname ... \endcsname} making one macro name out of it. This uses up only one token in \TeX{}'s main memory (and some string memory but this is comparatively cheap).

As an example take the normal \emph{shape} of \emph{series} roman in the computer modern \emph{family}, i.e. the cmr fonts. We define a macro \texttt{\cm@roman@normal} whose replacement text contains a single token containing all necessary information in its name. This macro itself is undefined.
\begin{verbatim}
\expandafter\edef\csname cm@roman@normal\endcsname{%
\expandafter\noexpand\edef\csname cm@roman@normal\endcsname{%}
<\langle I\rangle cmr5\%
<\langle VI\rangle cmr6\%
<\langle VII\rangle cmr7\%
<\langle VIII\rangle cmr8\%
<\langle IX\rangle cmr9\%
<\langle X\rangle cmr10\%
<\langle XI\rangle cmr@10 at 10.95pt\%
<\langle XII\rangle cmr12\%
<\langle XIV\rangle cmr12 at 14.4pt\%
<\langle XVII\rangle cmr17\%
<\langle XX\rangle cmr17 at 20.736pt\%
<\langle XXIV\rangle cmr17 at 24.8832pt\%
\end{verbatim}
\end{verbatim}
\end{verbatim}
The first \texttt{\expandafter} is needed because the macro name consists of / characters and we use \texttt{\csname ... \endcsname} to build them into the macro name. We then use \texttt{\edef} so that the second \texttt{\csname ... \endcsname} combination is expanded at definition time. Finally we need the \texttt{\expandafter \noexpand} trick to ensure that the resulting (undefined) macro is not expanded.

\subsection{Fonts for math}
To set up fonts for math one has to set up several assignments of the form
The actual math font assignments are carried out by the macro \select@group which is called only if the user selects the alphabet inside a formula. In this way, fonts not used in a certain document are not loaded, thereby saving space and time.

For every math alphabet, there must obviously exist at least one version. But it is perfectly legal that certain alphabets are available only in certain versions. Therefore we need a way to warn the user if he selects a version of an alphabet that does not exist.

If a math alphabet does not exist in a certain version the corresponding part of the version macro will look like

\def(math alphabet identifier){% 
\no@version@warning(version)  
(math alphabet identifier)}% 

which leads to a warning message if the alphabet is selected in this version.

A minute ago we promised to tell how we obtain the script and scriptscript sizes for a given text size. For every size and math group, you need a \textfont, a \scriptfont, and a \scriptscriptfont. The math fonts have to be switched for every size change. Since the math group assignments have to be in effect when the current math formula ends we make them all global. But then the old assignments must be restored at the end of the current group. This is done by inserting a macro call with the aftergroup primitive. The current text size is always available to this macro in the expansion of \f@size. The corresponding script size and scriptscript size (specified via \define@mathsizes), however, must be recorded somewhere. We use the following scheme for this: for every size s we define a macro \S@s (e.g. \S@XX) that globally defines two macros \sf@size and \ssf@size to expand to the corresponding script size and scriptscript size. With the help of these macros the right sizes can be extracted easily. Take for example size ‘lo’, with script size ‘7’ and scriptscript size ‘5’. The corresponding macro looks like

\expandafter\def\cname S@10\endcname  
{\gdef\sf@size(7)\gdef\ssf@size(5)
4.3 Special considerations

There are two special cases we must take care of. Both have to do with size changes within an alignment. Why is this special? The first problem appears when the size change occurs in the last column of an alignment. The token saved by the \aftergroup primitive will be inserted just after the \cr has been read. More precisely: after the end of the alignment template. But here only \noalign or the end of the alignment is allowed, everything else starts a new column. There is a simple fix for this: in the template of the alignment the hash mark (#) denoting the last column must be wrapped in a group. The same problem shows up if a size change occurs inside a \noalign.

5 Preliminary macros

As always we begin by identifying the latest version of this file on the VDU and in the log file.

\immediate\write\sixt@@n{File: 'fam.tex'
\fileversion \space \brate <\filedate> (FMI and RmS)}
\immediate\write\sixt@@n{English Documentation \brate <\docdate> (RmS and FMI)}

Following are a number of macros that will be used later.

\@spaces We define \@spaces to be an abbreviation for five space tokens.
\def \@spaces{space\space space\space space\space space}
This is also defined in latex.tex, but this code cautiously does not assume that any macros are defined elsewhere (except those in plain.tex).

\@gobble The \@gobble macro is used to get rid of its argument.
\def \@gobble{ }

\@empty The \@empty macro expands to nothing and is used to test for empty replacement texts.
\def \@empty{}

\@height The \@height, \@depth and \@width macros are made to conserve token memory.
\def \@height{height}
\def \@depth{depth}
\def \@width{width}

\fontwarning We need a macro that prints a warning message. We write to output stream 16 which means that the message will appear both in the transcript file and on the terminal.
\def \fontwarning{\immediate \write \sixt@@n {Warning: #1.}}

\@nomath \@nomath is used by all macros that should not be used in math mode.
\def \@nomath{ }

\no@version@warning The macro \no@version@warning is called whenever the user requests a math alphabet that is not available in the current version. The first argument is the name of the version (as a sequence of characters), the second is the control sequence that identifies the math alphabet. The \relax at the beginning is necessary to prevent \TeX from scanning too far in certain situations.
\def \no@version@warning{\relax\ifmode \fontwarning{No ' #1' version for math alphabet identifier \string#2}\fi}
We have to redefine one plain \TeX{} macro: We must remove \texttt{\textbackslash outer} from definition of \texttt{\textbackslash newfam} so that it can be used inside other macros. We also give a new name to \texttt{\textbackslash newfam} and \texttt{\textbackslash fam} to avoid verbal confusion (see the introduction).\footnote{For the same reason it seems advisable to \texttt{\textbackslash letfam} and \texttt{\textbackslash newfam} equal to \texttt{\textbackslash relax}, but this is commented out to retain compatibility to existing style files.}

\begin{verbatim}
def\newmathgroup\{alloc\#8\group\chardef\sixt\@n\}
\let\group\fam
%\let\newfam\relax
%\let\fam\relax
\end{verbatim}

\section{Macros for setting up the tables}

\texttt{\textbackslash newfontshape} Since this kind of definition is needed several times we provide a macro \texttt{\textbackslash newfontshape} that does the work for us.

\begin{verbatim}
def\newfontshape\#1\#2\#3\{\expandafter
  \edef\csname\#1\#2\#3\endcsname{\expandafter\noexpand
    \csname\#1\endcsname\#4\endcsname}
\end{verbatim}

\texttt{\textbackslash extra@def} The ‘extra’ macro is defined as follows.

\begin{verbatim}
\expandafter\def\csname extra\endcsname endcsname\#1\{%
  \hyphenchar\#1\m@ne\}
\end{verbatim}

We provide an abbreviation for this:

\begin{verbatim}
def\extra@def\#1\#2\#3\{
  \expandafter\def\csname extra\#1\#2\#3\endcsname{\endcsname}
\end{verbatim}

so that the above definition looks like

\begin{verbatim}
def\extra@def\{cm\{typewriter\}\{hyphenchar\\m@ne\}
\end{verbatim}

However, this is inefficient if there is nothing to do (i.e. if the third argument is empty), so we provide a special test for this case. Here is the actual definition:

\begin{verbatim}
def\extra@def\#1\#2\#3\{
\let\@tempa\@gobble
\ifx\@tempa\@gobble
  \expandafter\let\csname extra#1#2#3\endcsname\@gobble
\else \expandafter\def\csname extra#1#2#3\endcsname\#1<#3\i\}
\end{verbatim}

We store the argument \#3 in a temporary macro \texttt{\textbackslash @tempa}. This must have one parameter since \#1 is allowed in the third argument of \texttt{\textbackslash extra@def} (otherwise \TeX{} will not accept the definition).

\begin{verbatim}
def\@tempa\#1\#3\{
\end{verbatim}

We compare \texttt{\textbackslash @tempa} with a macro with one argument and empty replacement text, i.e. with \texttt{\textbackslash gobble}. If these two are the same, we \texttt{\let} the ‘extra’ macro equal \texttt{\textbackslash gobble}.

\begin{verbatim}
\ifx\@tempa\@gobble
  \expandafter\let\csname extra\#1\#2\endcsname\@gobble
\else \expandafter\def\csname extra\#1\#2\endcsname\#1<#3\i\}
\end{verbatim}

Otherwise, we build a definition.

\begin{verbatim}
\else \expandafter\def\csname extra\#1\#2\endcsname\#1<#3\i\}
\end{verbatim}

\texttt{\textbackslash preload@sizes} As we already explained, the macro \texttt{\textbackslash preload@sizes} provides a convenient way to specify fonts to be preloaded. It takes four arguments and its definition is as follows:

\begin{verbatim}
def\preload@sizes\#1\#2\#3\#4\{
\end{verbatim}

We define a macro \texttt{\textbackslash next}\footnote{We cannot use \texttt{\textbackslash @tempa} since it is needed in \texttt{\textbackslash pickup@font}.} that grabs the next \texttt{size} and loads the corresponding font. This is done by delimiting \texttt{\textbackslash next’s} only argument by the token \texttt{,} (comma).

\begin{verbatim}
def\next\#1\{
\end{verbatim}
The end of the list will be detected when there are no more elements, i.e. when \next's argument is empty. The trick used here is explained in Appendix D of the \TeXbook: if the argument is empty, the \if will select the first clause and \let \next equal to \relax. (We use the > character here since it cannot appear in font file names.)

\begin{verbatim}
\if>\#1
  \let\next\relax
\else
\fi
\end{verbatim}

Otherwise, we define \fontQname appropriately and call \pickupQfont to do the work. Note that the requested \textit{family/series/shape} combination must have been defined, or you will get an error.

\begin{verbatim}
\edef\font@name{\csname#l/#2/#3/##l\endcsname}%
\pickupQfont
\fi
\end{verbatim}

Finally we call \next again to process the next size. If \next was \let equal to \relax this will end the macro.

\begin{verbatim}
\next\%
\end{verbatim}

We finish by reinserting the list of sizes after the \next macro and appending an empty element so that the end of the list is recognized properly.

\begin{verbatim}
\next#4,
\end{verbatim}

We need a switch to decide if we have to change math fonts. For this purpose we provide \ifdefmathfonts that can be set to true or false by the \SQ macros, depending on whether math fonts are provided for this size or not. The default is, of course, to switch all fonts.

\begin{verbatim}
\ifdefmathfonts\define@mathfontstrue
\define@mathsizes takes the text size, script size, and scriptscript size as arguments and defines the right \SQ macro. (\define@mathfontstrue might be omitted if math fonts are to be defined for every size.)
\begin{verbatim}
\def\define@mathsizes#l#2#3{\expandafter\def\csname S@#l\endcsname{\gdef\sf@size{#2}\gdef\ssf@size{#3}\let\define@mathfontstrue}}
\end{verbatim}
\end{verbatim}

\begin{verbatim}
\ifdefnomathsize\define@nomathsizes#1{\expandafter\let\csname S@#l\endcsname \ifdefnomathfontsfalse}
\end{verbatim}

\section{Selecting a new font}

\subsection{Macros for the user}

\begin{verbatim}
\family
\series
\shape
\f@family
\f@series
\f@shape
\size
\f@size
\setnew@baselineskip
\end{verbatim}

As we said in the introduction, a font is described by four parameters. We first define macros to specify the desired \textit{family}, \textit{series}, or \textit{shape}. These are simply recorded in internal macros \f@family, \f@series, and \f@shape, resp. We use \edef's so that the arguments can also be macros.

\begin{verbatim}
\def\family#1{\edef\f@family{#1}}
\def\series#1{\edef\f@series{#1}}
\def\shape#1{\edef\f@shape{#1}}
\end{verbatim}

We also define a macro that allows specification of a size. In this case, however, we also need the value of \baselineskip. We cannot set \baselineskip immediately, so it is recorded in the macro \setnew@baselineskip. We use \edef here because the second argument (\#2) might be a macro.

\begin{verbatim}
\def\size#2{%
  \edef\f@size{#1}\
  \edef\setnew@baselineskip{\baselineskip \#2\relax}}
\end{verbatim}
The macro \selectfont is called whenever a font change must take place.

Its first action is to determine if the new font has the same size as the previous one. Here the macro \glb@currsize holds the current font size. Its expansion text may also be empty which means that we do not know what the current size is. As its name indicates, it is always set globally.

\ifx \glb@currsize \f@size

If the size is to be changed we must also change \baselineskip and a number of other parameters. This is done by the macro \glb@settings.

\else \glb@settings

Since these changes are done globally, we must ensure that the old values are restored at the end of the current group. We use \TeX's \aftergroup primitive to call \glb@settings again just after the current group ends. And that's all of special code for a size change.

\aftergroup\glb@settings \fi

We now generate the internal name of the font by concatenating family, series, shape, and current size, with slashes as delimiters between them. This is much more readable than standard \LaTeX's \twfbf, etc.

\edef\font@name{%
\csname\f@family/\f@series/\f@shape/\f@size\endcsname\}%

We call the macro \pickup@font which will load the font if necessary.

\pickup@font

Finally, we select the font. This finishes the macro \selectfont.

\font@name

\mathversion

\mathversion takes the math version name as argument, defines \math@version appropriately and switches to the font selected, forcing a call to \glb@settings if the version is known to the system.

\def\mathversion#1{%\ifx\csname#1\endcsname\relax
\font@uaxning{The requested version '#1' is unknown}\else
\def\math@version{#1}\glb@settings\aftergroup\glb@settings\fi}

7.2 Macros for loading fonts

\pickup@font

The macro \pickup@font which is used in \selectfont is very simple: if the font name is undefined (i.e. not known yet) it calls \define@newfont to load it.

\def\pickup@font{%
\expandafter\ifx\fontename\relax
\define@newfont
\else
\define@newfont\relax
\fi}

\split@name

\pickup@font assumes that \font@name is set but it is sometimes called when \f@family, \f@series, \f@shape, or \f@size may have the wrong settings (see, e.g., the definition of \getanddef@fonts). Therefore we need a macro to extract font family, series, shape, and size from the font name. To this end we define \split@name which takes the font name as a list of characters with backslash at the beginning delimited by the special control sequence \\nil. This is not very complicated: we first ensure that / has the right \catcode

\catcode'/'=12

and define \split@name so that it will define our private \f@family, \f@series, \f@shape, and \f@size macros.

\gdef\split@name#1/#2/#3/#4\nil{%\def\f@family{#1}}
\begin{verbatim}
\def\define@newfont{%
We have already mentioned that the token list that \splitQname will get as argument must not start with a backslash. To reach this goal, we will set the \escapechar to -1 so that the \string primitive will not generate an escape character. But then we must save \escapechar's current value. We use count register \count@ for this purpose.
\count@\escapechar
\escapechar\m@ne
Then we extract family, series, shape, and size from the font name. Note the four \expandafters so that \fontQname is expanded first, then \string, and finally \splitQname.
\expandafter\expandafter\expandafter\expandafter\split@name\expandafter\string\fontBname\@nil
If the family/series/shape combination is not available (i.e. undefined), we call the macro \wrongOfontshape to take care of this case. Otherwise, \extractQfont will load the external font for us.
\expandafter\ifx\csname\fOfamily/\f@series/\f@shape\endcsname\relax
\series\default@series
\fi
If this is still undefined, try the default family. Otherwise give up.
\expandafter\ifx\csname\fOfamily/\f@series/\f@shape\endcsname\relax
\family\default@family
\fi
At this point a valid family/series/shape combination must have been found. We inform the user about this fact.
\font@warning{Using 'f\f@family/\f@series/\f@shape' instead}%
\end{verbatim}
If we substitute a family/series/shape combination by the default, we don’t want the warning to be printed out whenever this (unknown) combination is used. Therefore we globally $\texttt{let}$ the macro corresponding to the desired combination equal to its substitution. This requires the use of four $\texttt{expandafter}$’s since $\texttt{\csname...\endcsname}$ has to be expanded before $\texttt{\@tempa}$ (i.e. the requested combination), and this must happen before the $\texttt{\let}$ is executed.

\begin{verbatim}
\global\expandafter\expandafter\expandafter\expandafter\let\@tempa
\csname\f@family/\f@series/\f@shape/\f@size\endcsname
\end{verbatim}

Now we can redefine $\texttt{\font@name}$ accordingly.

\begin{verbatim}
\edef\font@name{\csname\f@family/\f@series/\f@shape/\f@size\endcsname}%
\end{verbatim}

The last thing this macro does is to call $\texttt{\pickup@f ont}$ again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

\begin{verbatim}
\pickup@f ont
\end{verbatim}

\texttt{\strip@prefix} In $\texttt{\extract@font}$ we will need a way to recover the replacement text of a macro. This is done by the primitive $\texttt{\meaning}$ together with the macro $\texttt{\strip@prefix}$ (for the details see appendix D of the \TeX{}book, p. 382).

\begin{verbatim}
\def\strip@prefix#1>{}
\end{verbatim}

$\texttt{\extract@font}$ Here it comes: the macro solving all our problems (well, nearly all). What must this macro do? This is explained best with an example. Assume that $\texttt{family}$ is ‘cm’, $\texttt{series}$ is ‘sansserif’, $\texttt{shape}$ ‘normal’, and $\texttt{size}$ ‘12’. Assume further that this combination is defined, i.e. there exists the macro $\texttt{cm/sansserif/normal}$. (Otherwise $\texttt{\extract@font}$ doesn’t get called.) Its replacement text consists of one (undefined) control sequence looking like

\begin{verbatim}
<10>cmss10<12>cmss12<17>cmss17
\end{verbatim}

For reasonable styles one usually needs more sizes but this is sufficient to get the flavour. We will define a macro $\texttt{\extract@fontinfo}$ to find the external font name (‘cmss12’) for us:

\begin{verbatim}
\def\extract@fontinfo#1<12>#2#3<#4\nil{%
  \global\font\cm/sansserif/normal/12#2}
\end{verbatim}

so that when it gets called via

\begin{verbatim}
\expandafter\extract@fontinfo
  \string<10>cmss10<12>cmss12<17>cmss17\nil
\end{verbatim}

#1 will contain all characters before <12>, #2 will be exactly cmss12, and #3 will be 17>cmss17. The expansion is therefore

\begin{verbatim}
\global\font\cm/sansserif/normal/12 cmss12
\end{verbatim}

which is exactly what we want.

But this is only part of the whole story. It may be that the size requested does not occur in the $\texttt{cm/sansserif/normal}$ macro. And the simple definition of $\texttt{\extract@fontinfo}$ we gave above does not allow us to specify the font substitution that we explained in 3.1.

Both problems are solved with the same trick: We define $\texttt{\extract@fontinfo}$ as follows:

\begin{verbatim}
\def\extract@fontinfo#1<12>#2#3#4\nil{%
  \global\font\cm/sansserif/normal/12
  \ifcase 0#2#3\relax\or
  #3 \font@warning{Size 12 not available}
\end{verbatim}
How does this work? The first difference from the previous definition is that the characters of the external font name are split between parameters #2 and #3, #2 receiving only the first character. If this first character is not a digit, the \ifcase will get the 0 and select the first alternative. #2 and #3 are combined again and used as a file name. If #2 is a digit then the expansion of \ifcase will combine the 0 and #2 to a number.\footnote{Recall that 01 is a valid \textit{number} for \TeX.} Cases 1 and 2 select the second and third alternatives that use #3 as the substitution font.

The default case is reserved for a size that cannot be found in the tables. We achieve this by calling \texttt{\extractQfontinfo} via

\begin{verbatim}
\expandafter \extractQfontinfo
 \string\<10>cmss10<12>cmss12<17>cmss17
 <12>3\@nil
\end{verbatim}

If the size ('12' in this case) appears in the \texttt{\<10>...} macro everything works as explained above, the only difference being that argument #4 of \texttt{\extractQfontinfo} additionally gets the \texttt{<12>3} tokens. However, if the size is not found, everything up to the final \texttt{<12>} is in argument #1, #2 gets 3, and #3 and #4 are empty. Therefore the \texttt{\ifcase} will select the default alternative and write an error message.

We have cheated a bit, of course. Normally digits and characters like /<> are not allowed as part of control sequences. Additionally the macros are hidden inside other control sequences so that we have to build \texttt{\extractQfontinfo} in several steps. Putting everything together we define \texttt{\extractQfont} as follows.

\begin{verbatim}
\def\extractQfont{%
 \@tempa is made an abbreviation for the head of the definition of \texttt{\extractQfontinfo}.
 \def\@tempa{\def\extract@fontinfo####l)%
 Then we define \@tempb so that it expands to \texttt{<\textit{size}>}. We use this slightly complicated construction to ensure that all characters have \texttt{\catcode} 12. This is needed for the delimiter matching in macro expansion.
 \edef\@tempb{<\expandafter\strip@prefix\meaning\fontname>}%
 Now we can define \texttt{\extractQfontinfo}.
 \expandafter\extractQfontinfo\tempb##2##3<##4\@nil\%
 Remember that \texttt{\fontname} expands to the internal font name.
 \global\expandafter\font \fontname
 Here comes the \texttt{\ifcase}. For the benefit of the user, the warning messages are a bit more eloquent.
 \begin{verbatim}
 \ifcase0##2##3\relax\or
 \fontwarning{Font/shape ‘\f@family/\f@series/\f@shape’ in size \texttt{\tempb} space not available}%
 \fontwarning{Using ‘#3’ instead}\or
 \fontwarning{Font/shape ‘\f@family/\f@series/\f@shape’ not available}%
 \fontwarning{Using ‘#3’ instead}\else
\end{verbatim}
\end{verbatim}
There are two points to be explained here: \default@errhelp is the font to be selected if the requested size is not found in the tables. nofont@help denotes a token register that contains a help message for the user. Its definition is given below.

\begin{verbatim}
default@errfont \errhelp
ofont@help
\errmessage{Font \string\font@name\space not found}\%
\fi\%
\end{verbatim}

Now we must extract the font information from the family/series/shape macro. This is done in two steps: first generate the macro name by \csname...\endcsname and expand it to get its replacement text. Then use \string to convert this text into a sequence of character tokens with \catcode 12. We define \font@info to contain this sequence followed by \(<\text{size}>\) (which is stored in \@tempb).

\begin{verbatim}
\edef\font@info{\expandafter\expandafter\expandafter\expandafter\string
\csname \string\font@name\space\endcsname\@tempb}\%
\end{verbatim}

Now we call \extract@fontinfo. Note the 3<@nil tokens at the end.

\begin{verbatim}
\expandafter\extract@fontinfo\font@info 3<\@nil
\end{verbatim}

Finally we call the corresponding "extra" macro to finish things.

\begin{verbatim}
\csname extra/\f@family/\f@series \expandafter\endcsname \font@name \relax
\end{verbatim}

The \relax at the end needs to be explained. This is inserted to prevent T\TeX{} from scanning too far when it is executing the replacement text of the "extra" macro.

nofont@help

nofont@help is a token register containing a help message. It is defined using plain T\TeX{}'s \newhelp macro.

\begin{verbatim}
\newhelp
ofont@help
{You requested a font/series/shape/size combination that is totally unknown. I have inserted a special font name that will produce interesting effects in your output. There are two cases in which this error can occur: }
{1) You used the \string\size\space macro to select a size that is not available. }
{2) If you did not do that, go to your local wizard and complain fiercely that the font selection tables are corrupted!}
{And do not worry about the missing escape characters in the error traceback above!}
\end{verbatim}

8 Assigning math fonts to versions

\begin{verbatim}
definemathalphabet
\define@mathalphabet which is built to append definitions specific to a new math alphabet to the replacement text of a version macro. It takes six arguments: the math version name (as a string of characters), a control sequence identifying the new math alphabet, the number of the new math group (normally a control sequence defined via \countdef), and finally three strings of characters denoting font family, series, and shape. If the shape parameter (#6) is empty then the alphabet #2 is not available in version #1.

\begin{verbatim}
\edef\define@mathalphabet\(#2\#3\#4\#5\#6\)\%
\end{verbatim}

The first thing it does is to check if the name of the math version is already defined. This is the case if there already exist other math alphabets in this version. We must of course remember these definitions. To do so we save the contents of the macro in the token register \toks@.

\begin{verbatim}
\expandafter\ifx\csname #1\endcsname\relax
\end{verbatim}
If there is no other math alphabet in this version, we simply store an empty token list in this register.
\toks@C)%
Otherwise, we generate the control sequence denoting the macro using \csname...\endcsname and store its replacement text in \toks@. Note the three \expandafter primitives to achieve this.
\else
\toks@expandafter\expandafter\expandafter\csname #1\endcsname%
\fi
Depending on the shape parameter (#6) we have different things to do. We save the sequence of character tokens in a temporary control sequence.
\def\@tempa{#6}%
Now we globally redefine the version. Since the name of the version is given as a sequence of characters we must again build a macro name out of it. We use an \xdef so that the definition is expanded first.
\expandafter\csname #1\endcsname
This is necessary since we want to insert the contents of token register \toks@.
{\the\toks@}
Then we append the new definitions for the alphabet #1. The \noexpand is necessary to insert the (math alphabet identifier) without expanding it.
\def\noexpand#1%
We must now catch the case that the shape parameter #6 saved in \@tempa is empty, i.e. that the alphabet is not available in this version. We simply include a call to the \no@version@warning defined earlier.
\ifx\@tempa\@empty
{\noexpand\no@version@warning
 \noexpand\math@version
 \noexpand\#1\%}
Otherwise, we include a call to \select@group (see below) with the three arguments (math alphabet identifier), (math group number), and font family/series/shape definition macro.
\else
{\noexpand\select@group
 \noexpand\#1\%}
\fi%
Now the macro switching to the version #1 contains a definition for alphabet #1. Finally we force a call to \glb@settings at the next time the fonts change by globally redefining \glb@currsize.
\def\glb@currsize{}%
\define@mathgroup
\define@mathgroup is similar to \define@mathalphabet. This macro is never called when processing a document, only during the font definition phase (e.g. by a style file in \LaTeX{} or when dumping a format file). It is used for those math groups that are used via \mathchardef primitives. Since we don't need a (math alphabet identifier) to select those symbols, the macro takes only five arguments: the math version name as a sequence of character tokens, the (math group number) as a control sequence (it must already be allocated using \new@mathgroup) or as a digit (for groups 0 to 3, and font family, series, and shape name (as a sequence of character tokens). The first part is therefore completely analogous to the definition of \define@mathalphabet.
\def\define@mathgroup{#1#2#3#4#5{%
Since this code is never called by the user there is no need to issue a warning when the \textit{(math group number)} isn’t allocated. However, the font tables must be defined consistently! Instead of \texttt{\selectQgroup} it uses \texttt{\getanddefineQfonts} which has only two arguments: \textit{(math group number)} and the font \textit{family/series/shape} combination.

\begin{verbatim}
\getanddefineQfonts has two arguments: the \textit{(math number number)} and the \textit{family/series/shape} name as a control sequence.
\end{verbatim}

\begin{verbatim}
\def\getanddefine@fonts#1#2{\edef\font@name{\csname \string#2/\f@size\endcsname}}
\end{verbatim}

Then we call \texttt{\pickupOfont} to load it if necessary. We remember the internal name as \texttt{textfont@name}.

\begin{verbatim}
\def\getanddefine@fonts#1#2{\edef\font@name{\csname \string#2/\sf@size\endcsname}}
\end{verbatim}

Then we append the new \texttt{textfont}... assignments to the \texttt{math@fonts}.

\begin{verbatim}
\edef\math@fonts{\math@fonts \textfont#1\textfont@name \scriptfont#1\scriptfont@name \scriptscriptfont#1\font@name)}
\end{verbatim}

\selectQgroup has three arguments: the new \textit{(math alphabet identifier)} (a control sequence), the \textit{(math group number)}, and the \textit{family/series/shape} definition macro name. We first check if we are in math mode.

\begin{verbatim}
\def\select@group#1\select@group\ifmmode
\end{verbatim}

We do these things locally:

\begin{verbatim}
\let\math@fonts\@empty \escapechar\m@ne
\getanddefine@fonts#2#3%
\end{verbatim}

\footnote{Terrible harm will come to you if you don’t do it right! A crowd of angry users might come to stone you!}

\footnote{One might ask why this expansion does not generate a macro name that starts with an additional \texttt{\ character}. The solution is that \texttt{\escapechar} is set to \texttt{-1} before \texttt{\getanddefine@fonts} is called.
We globally select the math fonts...
\globaldefs\one \math@fonts

... and close the group to restore \globaldefs and \escapechar.
\egroup

As long as no size or version change occurs, the (math alphabet identifier) should simply switch to the installed group instead of calling \selectgroup unnecessarily. So we globally redefine the first argument (the new (math alphabet identifier)) to expand into a \group switch and then select this alphabet. Note that this redefinition will be overwritten by the next call to a version macro.
\gdef#1\{(\group #2)\#1%
If we are not in math mode nothing needs to be done.
\fi}

\glb@settings
The macro \glb@settings globally selects all math fonts for the current size. The first thing it does is to open up a group.
\def\glb@settings{\begingroup
This is done to keep the following changes local: set the \escapechar to -1 and make \math@fonts to expand to nothing.
\escapechar \m@ne
\let\math@fonts\@empty
Why do we \let \math@fonts equal to \@empty at this point? When \glb@settings gains control, a size change was requested and all previous font assignments need to be replaced. Therefore the old values of the fonts are no longer needed. For every group the new assignments are appended to \math@fonts. Now we set the script size and scriptscript size.
\csname S@f@size\endcsname
This also sets the define@mathfonts switch. If it is true, we must switch the math fonts. We execute the macro for the current math version. This sets \math@fonts to a list of \textfont... assignments.
\ifdef\math@fonts \csname \math@version \endcsname \fi
Then we set \globaldefs to 1 so that all following changes are done globally.
\globaldefs\one
The math font assignments recorded in \math@fonts are executed, \glb@currsize is set to the wanted \f@size, and the \baselineskip parameter is set accordingly by the macro \setnew@baselineskip and then multiplied by \baselinestretch.
\math@fonts
\let \glb@currsize \f@size
\setnew@baselineskip
\baselineskip\baselinestretch\baselineskip
Then we set the \strutbox and \normalbaselineskip.
\setbox\strutbox\hbox{\vrule@height.7\baselineskip
\@depth.3\baselineskip \@width\z@}%
\normalbaselineskip\baselineskip
The macro ends by closing the group. This restores all parameters changed locally (including \globaldefs!) to their previous values.
\endgroup}

\baselinestretch
In \glb@settings we used \baselinestretch as a factor when assigning a value to \baselineskip. We use 1 as a default (i.e. no stretch).
\def\baselinestretch{1}
The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

### Symbols

<table>
<thead>
<tr>
<th>Command</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>@depth</td>
<td>227, 237</td>
</tr>
<tr>
<td>@empty</td>
<td>227, 235-237</td>
</tr>
<tr>
<td>@gobble</td>
<td>227, 228</td>
</tr>
<tr>
<td>@height</td>
<td>227, 237</td>
</tr>
<tr>
<td>@nomath</td>
<td>227</td>
</tr>
<tr>
<td>@spaces</td>
<td>227, 234</td>
</tr>
<tr>
<td>@width</td>
<td>227, 237</td>
</tr>
</tbody>
</table>

### A

- \aftergroup ........................................... 230

### B

- \baselineskip .................................. 223, 229, 237
- \baselinestretch ................................... 237, 237

### D

- \defaultfamily ........................................ 229
- \defaultseries ........................................ 229
- \definealphabet ........................................ 229
- \define@mathgroup .................................. 224, 225, 234
- \define@mathgroup .................................. 224, 225, 235
- \definesizes .......................................... 225, 229
- \definemathsize ........................................ 225, 229
- \definemathsize ........................................ 225, 229

### E

- \errhelp ............................................. 234
- \errmessage .......................................... 234
- \escapechar ........................................ 231, 236, 237
- \extr@def ........................................... 224, 228
- \extract@font ......................................... 232

### F

- \f@family ............................................. 229
- \f@series ............................................. 229
- \f@shape .............................................. 229
- \f@size ................................................. 229
- \fam .................................................. 228
- \family ............................................... 229, 229
- \font@warning .......................................... 227

### G

- \getanddefine@fonts ................................. 226, 236
- \globcurssize ........................................ 230
- \globsettings .......................................... 237
- \globaldefs ............................................ 237
- \group ................................................. 224, 228

### I

- \ifdefmathfonts ...................................... 229

### M

- \math@version ......................................... 230
- \math@version ......................................... 229, 230

### N

- \new@font ............................................... 223, 228

### P

- \pickup@font .......................................... 230
- \preloadedizes ........................................ 224, 228

### S

- \scriptfont ........................................... 236
- \scriptscriptfont .................................... 236
- \select@group .......................................... 226, 236
- \selectfont ........................................... 223, 230
- \series ................................................. 223, 229
- \setnew@baselineskip ................................ 229
- \shape .................................................. 229, 229
- \size ................................................... 223, 229
- \split@name ............................................ 230
- \strip@prefix .......................................... 232
- \strutbox ............................................... 237

### T

- \textfont ............................................. 236

### W

- \wrong@fontshape ....................................... 231

---

◊ Frank Mittelbach  
Rainer Schöpf  
Fachbereich Mathematik  
Universität Mainz  
Staudinger Weg 9  
D-6500 Mainz  
Federal Republic of Germany  
Bitnet: schoepf@dmznat51