OFS: The Olšák’s Font System

The OFS is a \TeX macro for managing large sets of fonts. You can select the appropriate fonts comfortably by the names from font catalog used by a font foundry. It means you don’t have to remember short names of tfm files and/or short names of NFSS font families. The user interface of this macro is the same in \LaTeX and in plain but there are two independent implementations of this macro: first and more elaborate: based only on plain macros; second: based on NFSS macros for \LaTeX users.

If a text in this documentation is applicable only for the plain\TeX version of the OFS then the text is introduced by the word **\texttt{PLAINTEX}:**. If a text is applicable only for OFS version based on NFSS then it is introduced by the word **\texttt{LATEX}:**.

After the OFS macro is loaded, the following message is printed:

**\texttt{PLAINTEX}:** OFS (Olsak’s Font System) based on plain initialized. \texttt{<ver.>}

**\texttt{LATEX}:** OFS (Olsak’s Font System) based on NFSS initialized. \texttt{<ver.>}

Main features of the OFS:

- The user interface is the same for plain and for \LaTeX.
- You can use the \texttt{\fontusage} command which displays a short description of OFS macros in terminal and in the log file.
- You can use the real font names from font catalog.
- You can use the font divided into two \TeX metrics (basic and extended tfm) and they seem from the user’s point of view to be only one font. This is useful for fonts with more than 256 characters if you don’t want to use Omega.
- You can choose the \TeX internal encoding of fonts for your language in the beginning of your document. This feature is commonly used for Czech and Slovak languages: there are \TeX fonts which encode the alphabets of these languages by Cork (T1 encoding) or by ISO-8859-2 (IL2 encoding). T1 font encoding is common in \LaTeX world, whereas IL2 encoding is widely used in the Czech and Slovak \TeX community.
- The OFS defines the language of declaration files. These files define mapping of full names of fonts to **\texttt{PLAINTEX}:** tfm names or **\texttt{LATEX}:** NFSS short names of the font families.
- **\texttt{PLAINTEX}:** You can use individual variants of fonts in an independent manner in similar way as in NFSS (family, size, encoding, variant).
- **\texttt{PLAINTEX}:** You can declare different fonts for different sizes in the declaration files (usable for Computer Modern family first of all).
- **\texttt{PLAINTEX}:** The OFS includes support of the math fonts manipulation when the PostScript fonts and/or fonts at different sizes are used.
- Interactive macro \texttt{ofstex.tex} enables printing the paragraph samples in the chosen font families, printing the font table, font registers, samples of the mathematic and the lists of characters, including their \TeX sequences. All to do is to write \texttt{tex ofstest [allfonts]} or \texttt{csplain ofstest [allfonts]} on the command line and to follow the orders on the terminal.
1. The font families

The most current font families have the four commonly used variants: normal (\texttt{\rm})
bold (\texttt{\bf}), italic (\texttt{\it}) and bold italic (\texttt{\bi}). These variants are called “standard
variants” in OFS. After the font family is activated by \texttt{\setfonts} command (see below),
you can use the commands \texttt{\rm}, \texttt{\bf}, \texttt{\it} a \texttt{\bi} as a variant switches for the current
font family. The first three commands are well known in plain and the fourth command
switches into BoldItalic variant and it is introduced in the OFS.

Additional “nonstandard variants” can be declared in some font families and vice
versa some “standard variants” can be missing in other font families. Only the \texttt{\rm}
variant has to be present in all families.

The original names of fonts can be a little different from the names mentioned above
in some font families but the font switches names \texttt{\rm}, \texttt{\bf}, \texttt{\it}, \texttt{\bi} can be unchanged.
For example the family Helvetica has the variant “Oblique”, but we are still using the
\texttt{\it} switch for this variant.

If you want to use the font switches from another font families at the same time,
these switches can be declared by the \texttt{\fontdef} command (see below).

The font families are declared in the declaration files. These files have the similar
meaning in the plain\TeX as \texttt{fd} files in NFSS. The recommended suffix for the files are
\texttt{PLAINTEX}: \texttt{tex} (they map the family names to \TeX metrics) or \texttt{LATEX}: \texttt{sty} (they map
the family names to the NFSS short family names). These files can include more than
one font family declaration. The names of these files are chosen in order to you can
recognize which families are declared here. Examples:

\texttt{PLAINTEX}: \texttt{LATEX}:
\begin{verbatim}
sjannon.tex, sjannon.sty ... the big Jannon family by stormtype.com
a35.tex, a35.sty ... the basic 35 fonts by Adobe
\end{verbatim}

The user chooses from these files only such files needed by his/her document and
writes the names of these files into the header of the document. For more simplicity,
“global” files include the \texttt{PLAINTEX}: \texttt{\input} or \texttt{LATEX}: the \texttt{\RequirePackage} command
for the single declaration files. Examples:
\begin{verbatim}
skatalog.tex, skatalog.sty ... all fonts by stormtype.com
allfonts.tex, allfonts.sty ... all fonts at your \TeX installation
\end{verbatim}

2. The user interface

2.1. The header

For example, let us suppose that we will use fonts from Jannon and Dyna-
Grotesk family by Storm Type Foundry. Their declaration files are included in the
\texttt{storm} directory. You can get the \TeX metrics plus the OFS macro package from
www.cstug.cz/stormtype. You can buy the real fonts from Storm Type Foundry.
See www.stormtype.com for more details. We need to use \texttt{PLAINTEX}: \texttt{sjannon.tex},
\texttt{sdynamo.tex}; or \texttt{LATEX}: \texttt{sjannon.sty}, \texttt{sdynamo.sty}. declaration files. The first letter
s in their file name means that the fonts are made by Storm Type Foundry. We can
use the following line in the header of our document:

\begin{verbatim}
PLAINTEX: \input ofs [sjannon, sdynamo] % space before "[" is necessary
\end{verbatim}
Now we can use the families declared in sjannon and sdynamo declaration files. The \showfonts command writes a list of available families on terminal and to a log file. In the above case, the \showfonts lists the following text:

OFS (l.1): The list of known font families:

defaults:
- [CMRoman/] \rm, \bf, \it, \bi, \sl
- [CMSans/] \rm, \bf, \it, -
- [CMTypewriter/] \rm, - , \it, - , \sl
- [Times/] \rm, \bf, \it, \bi
- [Helvetica/] \rm, \bf, \it, \bi, \nrm, \nbf, \nit, \nbi
- [JannonAntikva/] \rm, \bf, \it, \bi, \mr, \mi
- [JannonText/] \rm, \bf, \it, \bi, \mr, \mi
- [JannonCaps/] \rm, \bf, \it, \bi
- [DynaGroteskDXE/] \rm, \bf, \it, \bi
- [DynaGroteskRXE/] \rm, \bf, \it, \bi
- [DynaGroteskLXE/] \rm, \bf, \it, \bi

The first 6 families are declared in OFS internally (you need not write any declaration file to use them). The next families are declared in specified declaration files.

Beside each family are listed option switches. The first four switches set “standard variants”. If a standard variant is not available then a dash sign is listed instead of the switch. The fifth and any other switches correspond to the “nonstandard variants”, if these exists. For example, the JannonAntikva and JannonText families have the extra variants medium and medium-italic (\mr and \mi switches are used here).

If you use the \fontusage command, then short description of OFS use is printed to the terminal and into the log file.

We mentioned the header in the form:

**LaTeX:** \usepackage [sjannon, sdynamo] {ofs}

**PlainTeX:** \input ofs [(file), (file), ...]

Instead of the header of this type you can also include definition files directly. In such case, the ofs.tex or ofs.sty do not have to be explicitly mentioned:

**LaTeX:** \usepackage{(file), (file), ...}{ofs}

**PlainTeX:** \input (file) \input (file) ...

Example:

**LaTeX:** \usepackage {sjannon} \usepackage {sdynamo}

We don’t recommend to mix the both variants of the header (in LaTeX specially).
2.2. The \setfonts command

\setfonts

Let us suppose that the sjannon and sdynamo declaration files were given in the header for the next examples. For example, the command:

\setfonts [JannonText/12pt]

sets the switches \rm, \bf, \it, \bi, \mr and \mi. These switches set variants of the JannonText family to 12pt font size.

The current variant used before \setfonts command is saved, but the family/size parameters of the current font are changed by this command. For example, if the BoldItalic variant for the TimesRoman was a current then the BoldItalic of JannonText family of size 12pt is current font set by \setfonts [JannonText/12pt]. If the new family has the current variant switch undeclared then the \rm variant is used.

The \setfonts makes all changes locally, so that T\TeX returns to the previous font and font family after the group is ended.

The parameters of the \setfonts[/⟨FamName⟩⟨size⟩] command can be empty: the \setfonts[/⟨FamName⟩] switches to the new font family but keeps the current font size and the \setfonts[⟨size⟩] changes the font size for all corresponded variant switches but keeps the current family unchanged. The command \setfonts[/] is syntactically correct but without any effect.

The CMRoman/10pt is default ⟨FamName⟩/⟨size⟩ after OFS for plainT\TeX is initialized.

The parameter ⟨FamName⟩, unless not empty has to be the same as the name of a family listed by the \showfonts. This parameter is case sensitive. If the parameter does not match any font family from given declaration files, the \setfonts command acts the same as the \showfonts command: all available families are listed. Thus, you can use the \setfonts [?] with the same effect as the \showfonts.

LATEX: The ⟨FamName⟩ parameter can be not only the “long-named” font family from \showfonts list, but the short name from NFSS can be used too. For example, the \setfonts[Times/] and the \setfonts[ptm/] has the same effect in LATEX.

PLAINTEX+LATEX: The parameter ⟨size⟩ can be written in more ways:

- ⟨number⟩ ... example: 12, 17.4,
- ⟨number⟩⟨unit⟩ ... example: 12pt, 17.4pt, 10dd,
- at⟨number⟩⟨unit⟩ ... example: at12pt, at17.4pt, at10dd,
- scaled⟨integer⟩ ... example: scaled1200, scaled\magstep3,
- mag⟨decimal number⟩ ... example: mag1.2, mag.7, mag2.0.

The first three possibilities have the same meaning. The keyword at is optional and if you omit the keyword and the unit, the at⟨number⟩pt is used. We can use the true... unit if we need not the relative unit associated by current \magnification factor: for example: 17truept. If the at keyword is present then the unit can’t be omitted—it means: at12 is not correct, but at12pt or simply 12 are correct values.

The scaled keyword has the same meaning as in T\TeX primitive \font. For example, if the design size of the font is 10pt (this is a common value) then scaled1200 is the same as at12pt.

The last possibility (keyword mag) is a new feature in OFS. The new font size is calculated from the current font size by multiplying by ⟨decimal number⟩. For example, by the command \setfonts[/12pt] followed by \setfonts[/mag2.] the current font
The size is changed to 24 pt. The decimal point is required in the \textit{(decimal number)}. Another example:

\begin{verbatim}
\def\small{\setfonts[/mag.7]}
The text \texttt{\small is smaller \small and smaller \small and more smaller}
and the normal size is used here.
\end{verbatim}

Note: the \texttt{(size)} parameter in \texttt{\setfonts} command changes only the font size but not the \texttt{\baselineskip}. The user has to do the \texttt{\baselineskip} setting by another way.

**LATEX:** The \texttt{\setfonts} sets the \texttt{\baselineskip} to the current value stored in NFSS. It means that if you want to change the \texttt{\baselineskip} register for your goal, you have to do it after \texttt{\setfonts} command in \LaTeX.

You can set only one variant of a font family (not the whole family) by the \texttt{\setfonts} command. This is done if the \texttt{"-\langle variant\rangle"} is appended to the \texttt{\langle FamName\rangle} parameter. In such case, the switches of the current family are not changed and only the new font is set. The \texttt{\langle variant\rangle} means the name of variant switch here. Examples:

\begin{verbatim}
\setfonts [JannonText-it/12] ...... sets the italics of the JannonText at 12 pt as the current font. The \texttt{\rm}, \texttt{\bf}, etc. are unchanged
\setfonts [JannonText-rm/] ......... sets the normal variant of the JannonText at the current size.
\setfonts [CMTypeWriter-sl/] ...... sets the cmsltt font as the current font at the current size.
\end{verbatim}

You can omit the family name even if the \texttt{"-\langle variant\rangle"} is present. The actual family is substituted in such case. Examples:

\begin{verbatim}
\setfonts [JannonText/12]
\setfonts [-bf/17] ... variant Bold of JannonText, size 17pt. Selectors \texttt{\rm}, \texttt{\bf}, \texttt{\it} and \texttt{\bi} keeps its original meaning: they switches between variants of JannonText in 12pt size. For example, the next command \texttt{\setfonts[Times/]} sets the Times family in 12pt size.
\end{verbatim}

BUT:

\begin{verbatim}
\setfonts [/17]\bf ... variant Bold of current family, size 17pt. Selectors \texttt{\rm}, \texttt{\bf}, \texttt{\it} and \texttt{\bi} switches in 17pt size now.
\end{verbatim}

**LATEX:** The text above about the keeping of original meaning of variant switches is not true in \LaTeX because it may break the NFSS philosophy. Thus, the commands \texttt{\setfonts [-bf/17]} and \texttt{\setfonts [/17]\bf} has the same meaning in \LaTeX.

### 2.3. The commands \fontdef and \addcmd

\fontdef \texttt{\fontdef \langle fontswitch\rangle [(\langle FamName\rangle)/(\langle size\rangle)]}

This declaration is roughly similar to

\begin{verbatim}
\gdef \langle fontswitch\rangle \{\setfonts [(\langle FamName\rangle)/(\langle size\rangle)]\}
\end{verbatim}
PLAINTEX: If the “–⟨variant⟩” is appended in ⟨FamName⟩ and the parameter ⟨size⟩ is not empty and it is not specified by mag keyword then the new declared control sequence \⟨fontswitch⟩ is not a macro but it is implemented by \global\font\⟨fontswitch⟩. It is called “fixed font” in an OFS terminology. The user can implement his/her native \⟨fontswitch⟩ by \fontdef without a knowledge about tfm filename.

LATEX: The declared \⟨fontswitch⟩ is always implemented as a macro including the \setfonts command. The reason is that the user access to native \⟨fontswitch⟩ is not simply possible in NFSS.

PLAINTEX+LATEX: You can type the exclamation mark “!” instead of ⟨FamName⟩ and the family current in the moment the \fontdef command was used is substituted. On the other hand the empty ⟨FamName⟩ means that the family current in the moment the \⟨fontswitch⟩ command was used is substituted. You can use the exclamation mark “!” instead of ⟨size⟩ parameter with the same meaning. Examples:

\setfonts [JannonAntikva]
\fontdef \small [7] % \small = \setfonts [7pt]
\fontdef \sffam [DynagroteskR] % \sffam = \setfonts [DynagroteskR/]
\fontdef \bigf [Times/17] % \bigf = \setfonts [Times/17pt]
\fontdef \ttfam [Courier/] % \ttfam = \setfonts [Courier/]
\fontdef \mylogo [Times-rm/mag.8] % \mylogo = \setfonts [Times-rm/mag.8]
% the fontsize will be always
% 0.8 times of the current size.
\fontdef \timbf [Times-bf/12] % \timbf = fixed-font, the same as:
% \global\font\timbf=ptmb8z at12pt
\fontdef \jansmall [!/7] % \jansmall = \setfonts [JannonAntikva/7]
\fontdef \janbi [!-bi/17] % \janbi = fixed-font, the same as:
% \global\font\janbi=sjnbi8z at17pt
\fontdef \tt [Courier-rm/!]
% \tt = fixed-font, the same as
% \global\font\tt=pcrr8u at10pt

The declaration of the \⟨fontswitch⟩ by \fontdef command is global but the \⟨fontswitch⟩ itself has a local effect in the place where it is used.

Since OFS version Oct. 2002, the \addcmd command is supported, which makes possible to concentrate whole font management in one place. The format of \addcmd is:

\addcmd \langlefontswitch⟩ {⟨commands⟩}

The meaning is same as

\def\langlefontswitch⟩ {⟨the original meaning of fontswitch⟩⟨commands⟩}

You can include new ⟨commands⟩ into the original content of the macro \⟨fontswitch⟩. The control sequence \⟨fontswitch⟩ has to be defined as a macro without parameters or as unexpandable control sequence (by \font, \chardef etc.) before \addcmd is used. The \addcmd redefines \⟨fontswitch⟩ as a macro without parameters in all cases. You can apply \addcmd on the same \⟨fontswitch⟩ more than once.

Examples:

\setfonts [JannonText/]
\fontdef \footnotefont [!/7]
\addcmd \footnotefont {\rm \baselineskip=9pt \relax}
\fontdef \sectionfont [!/12]
2.4. Test of a family name existence

**PLAINTEX**: You are able to test the font family declaration in your own marcos, it means whether the font is loaded from the file of declarations. The sequence \texttt{\knownfam (FamilyName)? iftrue} is used for such test. This sequence expands to \texttt{iftrue}, if the font family is declared otherwise expands to \texttt{iffalse}. The parameter \texttt{(FamilyName)} has to be brought in without the variant specification.

**PLAINTEX+LATEX**: By reason of the backward compatibility with the older version of OFS the \texttt{\ifknownfam [(FamilyName)]} does the same as \texttt{\knownchar} macro. Since the version Feb. 2004 of the OFS for plain it is recommended to use \texttt{\knownfam}, because of correct alignment of the primitives \texttt{\if*, \else, \fi}. LaTeX user can define \texttt{\knownfam} quite easily.

2.5. **LATEX**: OFS and NFSS

**LATEX**: This section is intended only for LaTeX users. The command \texttt{\OFSfamily [(FamilyName)]} converts the long family name to internal short NFSS family name. For example \texttt{\OFSfamily [Times]} expands to \texttt{ptm}. The macro works only on expand processor level thus we don’t get the error message if the \texttt{(FamilyName)} is not known. The \texttt{\OFSfamily} expands to the text “unknown” in such case. If you are using the \texttt{\OFSfamily} in your macro files and the NFSS try to substitute the unknown family then you can be sure that some misspelling occurs in \texttt{(FamilyName)} parameter or the required family is not known.

The example:

\begin{verbatim}
\usepackage [sjannon, sdynamo] {ofs}
\edef\rmdefault {\OFSfamily [JannonAntikva]}
\edef\sfdefault {\OFSfamily [DynaGroteskR]}
\edef\ttdefault {\OFSfamily [Courierir]}
\end{verbatim}

The meaning of the macros \texttt{\rmdefault, \sfdefault, \ttdefault} is described in the NFSS documentation.

OFS defines the command \texttt{\OFSfamilydefault [(FamilyName)]} which sets the basic family of the whole document. This family is used in the plain text and in the chapter headers etc. too (if the used class file is made by common LaTeX conventions). The command \texttt{\OFSfamilydefault} internally does:

\begin{verbatim}
\edef\familydefault {\OFSfamily [(FamilyName)]}
\end{verbatim}

and moreover it also cares for the case of the unknown \texttt{(FamilyName)}. If the \texttt{(FamilyName)} is not known then the list of the supported families is printed.

2.6. The font encoding

**LATEX**: The font encoding switching is the subject to the NFSS and OFS defines nothing more (i.e., packages \texttt{fontenc} and \texttt{inputenc} work as expected).
OFS sets the font encoding into CSfonts encoding by default. If you need to use fonts encoded in another encoding (T1 by Cork, for example) then you write \def\fotenc{8t} before the OFS is included and OFS will operate the fonts with this encoding. The name “8t” is an example of T1 encoding.

You can even switch the encoding inside the document:
\def\fotenc{8z} \setfonts[/] ... fonts in CSfont encoding  
\def\fotenc{8t} \setfonts[/] ... fonts in encoding by Cork

Moreover there are tools in OFS for correct macros expansion, that are dependent upon the font encoding (for example \v, \’, \ae).

By default, OFS set \loadingenc=0, which means, that font encoding change nor the command \setfonts does not change of the macros of the type \v, \ae. These macros hold their original plain meaning. Such a feature will welcome plain users, who dislikes too intelligent packages.

But, if the document starts with \loadingenc=1, for example:
\input ofs [a35,sjannon] \loadingenc=1

then \TeX checks during every run of the \setfont command, whether all necessary files named ofs-{encoding}.tex are loaded. These files contain macro definitions dependent on the preset font encoding. If these files are not loaded, \TeX loads them during the run of \setfonts. More informations on this topic can be found in the sections 3.3 to 3.5.

The OFS package contains three basic files with macros definitions dependent on the encoding: ofs-8z.tex, ofs-8t.tex and ofs-8c.tex. If you use another font encoding, you can create a new similar file. Commands \accentdef and \characterdef used in these files are described in details in the section 3.4.

### 2.7. The fonts in mathematics

**LATEX:** OFS for \TeX does nothing with the issue of math fonts. You need to use some \TeX package or build on the capabilities of NFSS.

**PLAINTEX** (to the end of this section): The command \setfonts and the control sequences declared by \fontdef switch fonts in text mode only. Until you use the \setmath command, all text between dollars is in Computer Modern in 10 pt/7 pt/5 pt sizes (text/index/index of index).

The \setmath command has the following syntax:

\setmath [(text size)/(index size)/(indexindex size)]

The parameters set sizes of mathematical fonts and they have same syntax as in the \setfonts command. The keyword mag means the magnification to the size of the current textual font. The empty parameter means the following substitution:

- text size: mag1.0
- index size: mag0.7
- indexindex size: mag0.5

It means that \setmath[//] has the same effect as \setmath[mag1./mag.7/mag.5].

The \setsimplemath command is defined in OFS as the equivalent of the \setmath[//] command.
The \texttt{\setmath} sets the mathematical fonts depending on the values of the macros \texttt{\fomenc} and \texttt{\mathversion}. The \texttt{\fomenc} means the mathematical encoding and the \texttt{\mathversion} means the version of the math-families set.

Mathematical encoding is set by the value of the macro \texttt{\fomenc}. There exist following possibilities:

- If the default value \texttt{\def\fomenc{PS}} (PostScript fonts) is used then \texttt{\setmath} sets the italic from the current font family as a mathematical italic (\texttt{\fam1}). It uses \texttt{\rm} from the current font family for digits and a some other symbols. The variant selectors \texttt{\rm}, \texttt{\it}, \texttt{\bf} and \texttt{\bi} work in math mode too. Moreover, when \texttt{\def\fomenc{PS}} is used then the math symbols from \texttt{\fam2} and Greek symbols (originally located in \texttt{\fam1}) are used from PostScript Symbol font. This font is much better visual compatible with most Postscript fonts than the Computer Modern symbols. Other symbols (e.g., big operators and big braces) stay in Computer Modern font because unfortunately there is no common alternative for these glyphs.
- If \texttt{\def\fomenc{CM}} is used then \texttt{\setmath} command keeps the Computer Modern fonts in math formulae. It sets only the desired sizes given in the parameters.
- It is possible to use \texttt{\def\fomenc{AMS}} after loading the file \texttt{amsfn.tex}. Mathematic symbols are the same as while using CM, but in addition you can use all mathematics symbols from AMSTeX.
- After loading the file \texttt{txfn.tex}, you can use two new encodings: \texttt{\def\fomenc{TX}} or \texttt{\def\fomenc{PX}}. In both cases the free TXfonts are used for mathematics, they are compatible with the font families Times and Helvetica. If you choose the value \texttt{TX} then pure TXfonts are used for all mathematic symbols, whereas the value \texttt{PX} means that TXfonts are going to be combined with italics and with the basic fontface of the actual font family (similar to \texttt{PS} encoding). OFS supports all control sequences of the mathematic symbols, as is mentioned in the TXfonts reference. There are all symbols from the AMSTeX and many more there. There are hunderts symbols from TXfonts.
- After loading the file \texttt{mtfn.tex}, it is possible to use \texttt{\def\fomenc{MT}}. Mathematics will contain italics and basic fontface of the actual font family as well. And it will be combined with the characters of the commercial version of the mathematics fonts MathTimes.

OFS supports two versions of math formulae as default: normal and bold version. The user can define another versions—see section 3.6. The current version is set by the contents of the \texttt{\mathversion} macro. You can say \texttt{\def\mathversion{normal}} or \texttt{\def\mathversion{bold}} before \texttt{\setmath} command. The \texttt{normal} version is used as default. Examples:

\begin{verbatim}
\setmath // $formula$ % formula in "normal" version
$\def\mathversion{bold}\setmath[/] formula$ % formula in "bold" version
\end{verbatim}
3. The inside of OFS for insiders

**LATEX**: All auxiliary macros of OFS are defined in \texttt{ofs.sty} file with the name \texttt{\ofs@macroname} in order to avoid the confusion with other style files. The macros used in declaration files have the name \texttt{\OFSmacroname}. Moreover OFS defines user-level macros \texttt{\fontdef, \showfonts, \fontusage, \rm, \bf, \it and \bi}.

**PLAINTEX**: All auxiliary macros of OFS are defined in \texttt{ofs.tex} file and they are listed in index at the end of this document. The convention with the $\emptyset$ character is not used because I personally hate this character in macro names.

3.1. Debugging

**LATEX**: Use the standard NFSS package \texttt{tracefnt} for tracing purposes.

**PLAINTEX**: There are four commands for tracing OFS:

\begin{itemize}
  \item \texttt{\nofontmessages} sets no tracing info.
  \item \texttt{\logfontmessages} sets the tracing info to log file only.
  \item \texttt{\displayfontmessages} sets the tracing info to log file and to terminal.
  \item \texttt{\detailfontmessages} sets the detailed tracing info to log and to terminal (all \texttt{\font} primitives are traced).
\end{itemize}

The \texttt{\logfontmessages} is initialized by default.

The warnings about unaccessible characters or encodings are always displayed on the terminal and they are written into the log file. If you want log output only, you can write \texttt{\let\displaymessage=\wlog}, because OFS uses this sequence for displaying messages on the terminal.

3.2. The robust and fragile commands

**LATEX**: LaTeX2e has its conventions to define robust commands. The command \texttt{\setfonts} and the \texttt{\langle\texttt{fontswitches}\rangle} declared by \texttt{\fontdef} are robust, of course. They can be used in texts for table of contents, indexes etc.

**PLAINTEX** (to the end of this section): Plain\TeX{} does not solve the problem of fragile commands and its users have their own solutions without any standardization. One solution is used in OFS.

What is a fragile command? We sometimes need to send some part of text to auxiliary file (for table of contents, index, etc.). We are doing it by \texttt{\write} primitive and in second run of \TeX{} this file is \texttt{\input}-ted. The problem is that the \texttt{\write} primitive prints the text to the file after all macro expansions and it may cause problems. For example, if the font switch is implemented as a complicated macro and it is used in \texttt{\write} parameter then the macro is stored in the file after its expansion. The error can occurs in most cases during the \texttt{\input} of such file. We say that the “fragile” command was used in \texttt{\write} parameter and that this command “was got spilt” in auxiliary file.

If the (potentially) fragile command defined in OFS is being sent to the file then the following message is printed on the terminal and to the log file during the \texttt{\input} of the file:

\texttt{ERROR !! The fragile command in the toc/ind/aux or similar file.}

You can solve this problem by the following steps:

1. Remove the auxiliary file with this command.
2. Include the following macro code into your document header:
3. Run TeX on your document again and again... See the OFS documentation for more info.

The fragile command in auxiliary file

You can follow this hint to remove the problem.

The detail explanation of this behavior follows. The \setfonts and other (potentially) fragile macros are implemented in OFS by the following way:

\def\macro {\%}
  \ifx\expandaction\noexpand
      \noexpand\macro
  \else
      \csname fragilecommand!\endcsname
  \fi
\}

\expandaction \fragilecommand! \fragilecommand \TeX run then the \fragilecommand runs and this command prints the message with the hint mentioned above.

If the user follows the hint then the \expandaction has the meaning of \noexpand when the \shipout is active (it means during the no-immediate \write parameter expansion). The \macro expands to \macro and this text is stored in auxiliary file.

The \shipout is not re-defined in OFS by default—only a suggestion is printed if fragile command in the \write parameter occurs. The reason is that a plain\TeX user may have his/her own redefinitions of \output routine or \shipout primitive thus OFS for plain does not do any redefinitions at this level itself. Unlike LaTeX users the plain user needs exactly to know how his macros work.

The code above illustrate the definition of an abstract macro \macro. The similar code is used for the following actual macros in OFS: \setfonts (section 2.2), \setmath (section 2.7), \fontdef macro (section 2.3), \setextrafont, \printcharacter, \printaccent (section 3.4), \accentabove and \accentbelow (section 3.6). If you use the hint above then these macros become “robust” in LaTeX sense of this word.

3.3. Declaration files

LATEX: The declaration files for OFS are common LaTeX style files which includes mapping from long family names to NFSS family names of one or more font families. These NFSS families are declared in common \fd files. Use NFSS documentation to create \fd files. You can use the following commands in the OFS declaration files:

- \OFSprocessoptions: This macro is undefined by default but it has the meaning \relax during ofs.sty file is scanned and its options are included. You can test by \ifeq the value of this control sequence in order to skip the \RequirePackage{ofs}.
\OFSextraencoding • \OFSextraencoding \{(extra encoding)\}: The macro stores the \{(extra encoding)\} into memory and does \input \{(extra encoding)\}.ini.def. We assume that the corresponding definitions for \{(extra encoding)\} are included in this file. See se1ini.def. This file includes the declarations for extra encoding SE1 for fonts by Storm Type Foundry. If the \{(extra encoding)\}.ini.def file was included already it is not included again. Attention: use uppercase letters for \{(extra encoding)\} parameter but use lowercase letters in filename.

\OFSputfamlist • \OFSputfamlist \{(text)\}: The macro adds the \{(text)\} into the list of family names. This list is printed by \showfonts command or if unknown family is used in \setfonts.

\OFSdeclarefamily • \OFSdeclarefamily \{(FamName)\} \{(NFSS-name)\}: This macro does an actual \OFSdeclarefamily mapping from \{(FamName)\} (long family name) to the \{(NFSS-name)\} (short NFSS family name). Moreover, it stores the line about this family name to the list of family names which is printed by \showfonts command.

\OFSnormalvariants • \OFSnormalvariants: This macro stores the list of standard switches \rm, \bf, \it, \bi into the list of family names which is printed by \showfonts command.

PLAINTEX (to the end of this section): The declaration files have the extension \text and we assume that there is a locking code in them so that the file will not be read twice. If the ofs.tex is not included already then it have to be included at the begin of declaration file.

You have to define the mapping from long family names to \text names in the declaration files. You can use the following commands:

\protectreading • \protectreading \{(filename)\}(space) — the flag about the \{(file)\} reading is saved in the memory. If the command is run with the same parameter once more, the \endinput is executed. It means that next declarations are protected against the multiple reading.

\ofsputfamlist • \ofsputfamlist \{(text)\}: The macro adds the \{(text)\} into the list of family names. This list is printed by \showfonts command or if unknown family is used in \setfonts.

\ofsdeclarefamily • \ofsdeclarefamily \{(FamName)\} \{(commands)\}: This macro declares the new font family with the name \{(FamName)\}. The \{(FamName)\} is stored into the list of family names which is printed by \showfonts command. If this family is used by \setfonts then the \{(commands)\} are performed. We assume that the \{(commands)\} include \loadtextfam command and (zero or more) \newvariant commands.

\loadtextfam • \loadtextfam: This macro loads four fonts with given metrics. See below for the syntax and the detail explanation of this command.

\newvariant • \newvariant\{(digit)\} \{(switch)\} \{(Variant)\} \{(space)\} \{(metric)\};\{(extra-enc)\};: This macro sets the “nonstandard” variant for given font family. See below for the detail explanation of this command.

\modifyenc • \modifyenc \{(encoding)\};\{(identifier)\}; — the exceptions are added with respect to the basic encoding, see section 3.5.

\fsize • \fsize: The information about the actual font size of the last selected font family is stored in this macro. The value of this macro can be in one of the two forms: at\{(dimen)\} or scaled\{(number)\}. This depends on the form of \{(size)\} parameter given in \setfonts command.

\fotenc • \fotenc: The actual encoding is stored in this macro. The common values are: 8z for encoding by CS-fonts (by ISO-8859-2) or 8t for encoding by Cork. The part of \text name (where encoding is specified) is recommended for values of \fotenc
macro. If \fotenc is undefined at the time of OFS is initializing, the OFS makes
\def\fotenc{8z} else the \fotenc is unchanged.

\extraenc
• \extraenc — macro, that stores the information about the extra encoding. This
information is copied from the parameter ⟨extra-enc⟩, that is located in the
\loadtextfam command.

\defaultextraenc
• \defaultextraenc — if you redefine this macro, the extra encoding of the basic
families and the families from a35.tex can be changed. The default value of this
macro is 8c.

\setfontshook
• \setfontshook: This macro is called from \setfonts macro before ⟨commands⟩
from \ofsdeclarefamily are performed.

\registrertfm
• \registrertfm ⟨symbolic name⟩ ⟨from⟩−⟨to⟩ ⟨real metric⟩: You can declare differ-
ent tfm names for different font sizes. See section 3.7 for more details.

\registerenc
• \registerenc ⟨FamilyName⟩: ⟨encoding⟩ ⟨space⟩ — enables the limitation of us-
age the font families only for certain encodings. See section 3.9.

The \loadtextfam command used in declaration files has the following syntax:

\loadtextfam ⟨Variant-rm⟩ ⟨space⟩ ⟨metric-rm⟩;%
  ⟨Variant-bf⟩ ⟨space⟩ ⟨metric-bf⟩;%
  ⟨Variant-it⟩ ⟨space⟩ ⟨metric-it⟩;%
  ⟨Variant-bi⟩ ⟨space⟩ ⟨metric-bi⟩;⟨extra-enc⟩;%

The percent characters at the ends of lines here mean that no spaces are allowed after
semicolons. You can save the long name of the used variant by ⟨⟨Variant-..⟩⟩ pa-
rameter. This name us used only when the OFS is traced by \logfontmessages or
others commands. The parameters “⟨⟨Variant-..⟩⟩ ⟨space⟩” are optional. If this pa-
rameter is omitted then default value is stored: rm: ( ), bf: (Bold), it: (Italic),
bi: (BoldItalic).

The parameters ⟨metric-..⟩ are the names of the tfm files for the appropriate variants.

The \loadtextfam command does roughly the following work:

\font\tenrm=⟨metric-rm⟩ \fosome
\font\tenbf=⟨metric-bf⟩ \fosome
\font\tenit=⟨metric-it⟩ \fosome
\font\tenbi=⟨metric-bi⟩ \fosome

We assume that all ⟨metric-..⟩ parameters are written with the \fotenc macro in order
to make the switching to others encodings possible.

The ⟨extra-enc⟩ parameter is the name of the extra encoding. If this parameter is
non-empty then the \loadtextfam command redefines temporally the \fotenc macro:
\def\fotenc{⟨extra-enc⟩} and it expands all parameters ⟨metric-rm⟩, ⟨metric-bf⟩,
⟨metric-it⟩ and ⟨metric-bi⟩ again. The results of these expansions are stored into mem-
ory. They are the “extra metrics” connected to the “basic metrics”. If the ⟨extra-enc⟩
parameter is empty then there are no “extra metrics” connected to “basic metrics”.
One can use a macro which can need the access to the extra metric concerned to the
basic metric of the current font. The macro for \euro character is the good example of
these needs.

Some ⟨metric-..⟩ (except of ⟨metric-rm⟩) can be omitted. When the ⟨metric-XX⟩ is
empty then the \loadtextfam command does roughly the following work:
\def\tenXX{\message{WARNING: the needed font variant is missings}}

13
It means that if the user needs the omitted variant then the message is printed to the log file and to the terminal and no font change is done.

The \setfonts command does not change the meaning of the macros \rm, \bf, \it and \bi. It only changes the font switches \tenrm, \tenbf, \tenit, and \tenbi respectively. The first three font switches are known in plain and the last one is introduced in OFS. The macros \rm, \bf, \it and \bi store the information about last selected variant into control sequence \currentvariant. This information has the form of the letter M (for \rm), F (for \bf), T (for \it) and I (for \bi). It is stored by \let\currentvariant=⟨letter⟩ because this code is not expanded. Thus we need not to implement a special “robust” code to the macros for variant switches.

Note that the \loadtextfam command sets the font switches \tenrm, \tenbf, \tenit and \tenbi to the fonts of arbitrary size given by the contents of the \fosize macro. The word “ten” in names of font switches is used only for the historical reasons and it does not mean that the font is loaded at 10 pt size.

You can object that the repetitive calls of \setfonts runs the font loading on the four fonts in given font family again and again. This can be time consuming operation. But you are not right. \TeX stores the font information from font loading in its internal memory and if the \font primitive is applied again to the same font then \TeX uses the information stored before and it needs not to load the font again.

If \loadingenc>0 the command \loadtextfam reads the file ofs-⟨enc⟩.tex before fonts loading. If the parameter ⟨extra-enc⟩ is non-empty, it loads moreover the file ofs-⟨extra-enc⟩.tex. These files are read only once. Empty lines and ends of lines are ignored during reading. Reading is performed inside the group. The character categories are locally set in accordance to plain \TeX (and \catcode'@=11). The \globaldefs=1 is set. It means, that all macros and values from the file ofs-⟨encoding⟩.tex are defined globally. That does not matter, because newly loaded encoding files do not conflict with the previous ones (see the commands \characterdef and \accentdef in section 3.4). It does not matter at all, if there are loaded more files, than is necessary at a given instant. It is not wise to read the encoding file repeatedly, when the command {...\setfonts...} is executed. This is the reason of the global definition. If the user dislikes the global predefined macros (he/she wants to add the article into the proceedings, where such a predefined macros can collide with other articles), he lets the \loadingenc=0. In this case, the declaration files have to be loaded manually by \input command at the beginning of the article.

You can declare a nonstandard variant in ⟨commands⟩ of the \ofsdeclarefamily by the \newvariant command. The \newvariant command does roughly the following:

\fsize
\let\currentvariant=⟨digit⟩ \ten⟨switch⟩

Moreover the \newvariant stores the “extended metric” connected to the “basic metric” if the ⟨extra-enc⟩ is not empty.

If the OFS needs to return to the last “nonstandard variant” then it does it by the value of the \currentvariant. If the new family has the “nonstandard variant” with the same ⟨digit⟩ as a previous family then this variant is used and OFS does not switch to the \rm variant. You can declare the variants of various families but the similar “type” with the same ⟨digit⟩. There are only ten digits thus we can distinguish only ten different “types” of “nonstandard variants”.

The macro \setfonts can change the meaning of the macros \loadtextfam and \newvariant if the ⟨variant⟩ is specified in ⟨FamName⟩ parameter of \setfonts. It is
We describe the operations of \texttt{\setfonts [\langle FamName\rangle]/\langle size\rangle]} command here in detail. This command calculates and defines the \texttt{\fosize} macro by the \texttt{(size)} parameter. If the \texttt{\langle FamName\rangle} is not empty and the \texttt{\langle variant\rangle} is not given then \texttt{\setfonts} performs \texttt{\def\currentfamily{\langle FamName\rangle}}. On the other hand, if the \texttt{\langle FamName\rangle} is empty then the \texttt{\currentfamily} is used for restoring the family name. If the \texttt{\langle variant\rangle} is given then \texttt{\setfonts} redefines the \texttt{\loadtextfam} and \texttt{\newvariant} macros at the temporary time. This behavior is described in previous paragraph. Then the \texttt{\setfonts} runs \texttt{\setfontshook} and \texttt{\langle commands\rangle} specified as a parameter of \texttt{\ofsdeclarefamily} of the appropriate \texttt{\langle FamName\rangle}. It also runs macro \texttt{\runmodifylist}, that at certain circumstances sets the exceptions from the chosen encodings (see section 3.5). Finally the \texttt{\setfonts} runs \texttt{\ignorespaces} at the end of its run in order to ignoring the possibly forgotten space after “]”.

## 3.4. The font encoding and the character declaration

\texttt{\setextrafont} \texttt{\font\extrafont=\langle extra metric connected to the current metric\rangle} \texttt{\extrafont}

\texttt{\fontencoding{\langle extra encoding\rangle}} \texttt{\selectfont}

\texttt{\extchar\langle number\rangle}.

\texttt{\characterdef} has the following syntax:

\begin{verbatim}
\characterdef \langle sequence\rangle \langle encoding\rangle \langle space\rangle \langle number\rangle
% example:
\characterdef \promile 8z 141
% or
\characterdef \langle sequence\rangle \langle encoding\rangle \langle space\rangle \langle\langle commands\rangle\rangle
% example:
\characterdef \promile 8t \{\%\char24 \}
\characterdef \promile * \{\vrule height1ex width1ex\relax\}
% in another encodings
\end{verbatim}

If the current encoding is the same as \texttt{\langle encoding\rangle} then \texttt{\langle sequence\rangle} will expand to the token of category 12 with the code \texttt{\langle number\rangle} or it expands to the \texttt{\langle commands\rangle}. All work
is done at expand processor level when \langle sequence \rangle is used. You can declare the same \langle sequence \rangle for more encodings, see the \texttt{promile} declarations in previous examples:

\def\fotenc{8z} \texttt{promile \% expands to the token with the code 141}
\def\fotenc{8t} \texttt{promile \% expands to the commands \%\char24}

Moreover you can simply declare the access to the extra encoding:

\characterdef \euro 8z 134
\characterdef \euro 6s 37

\def\fotenc{8z} \euro \% expands to the token of the code 134
\def\fotenc{8t} \euro \% expands to: {\setextrafont \langle sequence \rangle:--\langle extra-enc \rangle}

The second example is working only if the extra metric connected to the current metric exists (see \texttt{loadtextfam} and \texttt{newvariant} commands) and the extra metric has the \texttt{6s} encoding. If this is not valid then the \euro prints the warning about inaccessibility of the \euro character to the terminal and to the log file.

Now, we explain the behavior of the \characterdef macros in more details. The \characterdef command defines the \langle sequence \rangle as \texttt{\printcharacter{\langle sequence \rangle}}, it means that \texttt{promile} expands to \texttt{\printcharacter{promile}} and \euro to \texttt{\printcharacter{euro}} in our examples. If you use the \characterdef twice to the same \langle sequence \rangle then it does not matter because the definition is still the same. Moreover, \characterdef defines the special macro \langle sequence \rangle:--\langle encoding \rangle in order to this macro expands to the token of given \langle number \rangle code or to the given \langle commands \rangle. The more work is done by the \texttt{\printcharacter} macro. This macro checks if the \langle sequence \rangle:--\texttt{fotenc} is defined. If it is true then \texttt{\printcharacter} expands to the contents of this macro. Else \texttt{\printcharacter} checks if the extra metric is connected to the current font. If it is true then \texttt{\printcharacter} checks if the \langle sequence \rangle:--\langle extra-enc \rangle is defined where \langle extra-enc \rangle is the encoding of the extra metric. If it is true then \texttt{\printcharacter} expands to

\texttt{\setextrafont \langle sequence \rangle:--\langle extra-enc \rangle}

If all attempts fail then the \texttt{\printcharacter} try to print the default character independent on encoding. It means, the \texttt{\printcharacter} checks if the \langle sequence \rangle:* is defined and if true, it expands to this macro.

If this is false then the \texttt{\printcharacterwarn\{sequence\}} is run. The implicit value of this macro prints out a warning, that the character \langle sequence \rangle is not available. It is printed on the terminal and into the log file. No character is printed to \texttt{dvi} output.

If we want to omit the warning printing, we can redefine the \texttt{\printcharacterwarn} for example by following way:

\def\printcharacterwarn #1{?(#1)?}

The \characterdef does not redefine the defined control sequences since the version OFS Mar. 2004. It defines only sequences, that have the meaning \texttt{undefined} or \texttt{relax}. Otherwise (and also if the sequence is not defined by previous \characterdef) it prints the warning, that the definition is ignored. The reason is that the encoding files declares by \characterdef command enormous amount of new control sequences. But the programmer have not to know all of them. It is possible, that he/she uses the same name for his/her own macro. In this case, the \characterdef keeps the macro defined by the programmer and lets the appropriate character unaccessible. You have to write
If the current encoding is the same as \texttt{(encoding)} then \texttt{(sequence)} followed by \texttt{(char)} expands to the token of category 12 with the code \texttt{(number)} or to the \texttt{(commands)}. This work is done at expand processor level. If the declared \texttt{(char)} is * then the \texttt{(sequence)} expands to the token of given \texttt{(number)} code or to the given \texttt{(commands)} in the case of the actual \texttt{(char)} does not match with all declared \texttt{(chars)}.

The possibility of the use of the extra metric is the same in \texttt{accentdef}-ed macros as in the \texttt{characterdef}-ed macros.

Now, we explain the functionality of the \texttt{accentdef}-ed macros in more details. The \texttt{accentdef} command defines the \texttt{(sequence)} as a macro with one non separated parameter \texttt{#1} which expands to the \texttt{printaccent{(sequence)}\{#1\}}. For example, \texttt{\v{E} 8z} expands to \texttt{\printaccent{v}{E}}. Moreover, \texttt{accentdef} defines the macro \texttt{(sequence):(char):(encoding):(space):(number)} in order to this macro expands to the token of given \texttt{(number)} code or to the given \texttt{(commands)}. The more work is done by the \texttt{printaccent} macro. This command checks if the \texttt{(sequence):(char):-\fotenc} is defined. If it is true then \texttt{printaccent} expands to the contents of this macro. Else the \texttt{printaccent} checks if the extra metric is connected to the current font. If it is true and if this extra metric has \texttt{(extra-enc)} encoding then \texttt{printaccent} checks if the \texttt{(sequence):(char):-\fotenc} is defined. If it is true then \texttt{printaccent} expands to \texttt{\setextrafont (sequence):(char):-\fotenc}. Else \texttt{printaccent} checks if the macros \texttt{(sequence):*:\fotenc} and \texttt{(sequence):*:\fotenc} are defined in this order. If the first one is defined then \texttt{printaccent} expands to this macro and appends the \texttt{(char)}. If only the second one is defined then \texttt{printaccent} expands to:

\begin{verbatim}
\setextrafont (sequence):*:\fotenc (normalfont) (char)
\end{verbatim}

where \texttt{(normalfont)} is the font switch to the current font at the start of \texttt{printaccent} macro. If all attempts fail so far then \texttt{printaccent} try to use the macros \texttt{(sequence):(char):*:} or \texttt{(sequence):*:\fotenc (char) in this order. If all the previous commands fail, the \texttt{printaccentwarn{(sequence)}{(character)}} is run. The default value of this macro prints out on the terminal and into the log file the warning about the unaccessibility of the accented character and no character is printed on dvi output.
Note that the character from extra metric inside the word breaks the kerning around this character and breaks the possibility of hyphenation of this word. It is extremely recommended that a basic metric encodes all alphabet used in current language in order to minimize switching to extra metric. For example, the 8t and 8z encodings are good choice as basic metric for Czech and Slovak languages.

If we want to take out predeclared character (see so called \langle exceptions \rangle in the next section), we can use the commands \characterdel and \accentdel. These commands have to have the same parameters like \characterdef and \accentdef respectively and they take out the command definition \langle sequence \rangle:-(encoding) and \langle sequence \rangle:-(char):-\langle encoding \rangle respectively.

3.5. \textsc{plainTeX}: Macro files dependent on the encoding and encoding exceptions

Commands \characterdef and \accentdef described in the previous section redefines macros dependent on the encoding (\v, \ae, etc.). In this section, we are going to describe the conception of placement these macros.

Macros declarations by means of \characterdef and \accentdef have to be written into the files called ofs-(encoding).tex (so called encoding files). The command \loaddtextfam (called from the \setfonts macro) reads these declarations from these files while \loadingenc=1 is set.

Every encoding contains its own basic set of characters and accented types. This set is registered in the encoding file by the \characterdef and \accentdef commands. Particular font families can contain some additional characters or some characters can be missing in reference to that basic set. These exceptions are declared by means of the command \modifydef:

\modifydef \langle encoding \rangle:-(\langle identifier \rangle); \{\langle exceptions \rangle\}

The \langle exceptions \rangle contain the commands \characterdef, \accentdef, \characterdel and \accentdel. The \*del commands have to contain the same value of the character in the argument as in basic encoding set. If any character has to be redefined, the commands \*del and \*def corresponding to this character must be written one after another.

The command \modifyenc used in the parameter of the \ofsdeclarefamily macro is a “link to \langle exceptions \rangle”. You can mark by \modifyenc command that the family contains \langle exceptions \rangle with respect to the basic encoding set. The command has following notation:

\modifyenc \langle encoding \rangle:-(\langle identifier \rangle);%

You can list more of one command for every font family (these commands can contain different \langle encoding \rangle as well). Nothing is done, if this command links to \langle exceptions \rangle, that were not yet declared.

An example of the exceptions declaration 8z:csfonts can be found in the file ofs-8z.tex and links to them are used in families CM* in the file ofsddef.tex.

Commands \modifyenc are run at each \setfonts. As the matter of fact these commands only stores their parameters into so called “list of links” (to the macro \newmodifylist). At each start of the \setfonts, the new list of links is created. The \modifyenc command stores its parameter into this list only if \langle encoding \rangle is equal to \fotenc or \extraenc. The exception handling provides the command
The macro \runmodifylist compares the “list of links” of the previous family (\modifylist) with the “list of links” of the newly set family (\newmodifylist). The \runmodifylist finishes its activity, if both lists are the same or \modifylist has the meaning \relax. Otherwise, the setting of exceptions is run: At first, meanings of \characterdef\→\characterdel and \accentdef\→\accentdel are exchanged and \modifylist is run. In other words, the macros dependent on the encoding are returned to the initial state (without exceptions). During this activity, the deleting of the character is ignored, if the character was declared immediately before (see the rule about character redefinition above). Next, the command \runmodifylist returns the \characterdef and \accentdef into the initial state and run \newmodifylist. All the redefinitions, that takes place during this activity, are just local. The mechanism of two lists ensures, that for example:

\setfonts [Family1/] ... \setfonts [Family2/] ...

the exceptions of the actual encoding will be correctly set even for the Family2, even though the Family1 has another set of exceptions than Family2.

The control sequence \modifylist has the meaning of the empty macro after the OFS startup. The macro programmer can set \let\modifylist=\relax to override every set of exceptions. Note, that declaration commands \modifydef store (exceptions) into the memory and execute them in order to define all declared sequences corresponding to \printcharacter and \printaccent respectively. It means, that every control sequence from all exceptions is defined (the message \texttt{undefined control sequence} is not displayed). Moreover OFS has perfect view whether the control sequence is available or not in the actual family. The macro programmer can then redefine macros \print*warn.

The command \modifydef slightly changes commands \accentdef, \accentdel, etc. for a temporary time and then it executes the (exceptions). The control sequence \skipfirststep forbids the execution of the macros in the (exceptions) during the activity of \modifydef. It acts just like \relax, but during the execution of the (exceptions) by means of \modifydef the whole part of (exceptions) behind this sequence is omitted.

Identifiers (encoding):\lccodes and (encoding):\ienc are reserved for usage in the macros out of OFS. OFS does not consider the setting of the characters \lccode, \uccode. A macro package taking care of these values can properly define commands \lccodes and \lccodesloop and runs \csname (encoding):\lccodes\endcsname. These above mentioned commands are not defined in OFS at all. The declarations (encoding):\lccodes are locoted in the files ofs-8t.tex and ofs-8z.tex, eventhough they are not used in OFS. It bears upon the text fonts encoding. An example of (encoding):\lccodes usage is placed in the macro called lang.tex. Macro \texttt{incc.tex} uses (encoding):\ienc. More informations can be found in the appropriate documentation.

The declarations of the most commonly used exceptions are written directly into the encoding files. The declarations of the less usual exceptions (related only to some font families) can be written behind \endinput of the declaration files. You can use a command \modifyread in (commands) of ofsdeclarefamily:

\modifyread \{filename\};%
This command reads the file from the first appearance of the sequence \modifytext, but only if the \loadingenc is positive. It is suitable to place the \modifytext sequence behind \endinput, so the \modifyread command reads only that part of the file, which has not been read before. Assigning is global during the reading, the empty lines and ends of lines are omitted. The file is not loaded repeatedly.

This command gives you chance to concentrate the font families declarations and encoding exceptions declarations into the single file. \TeX reads the exceptions in the time it needs them, so the \TeX memory is spared. An example is placed in the file \texttt{slido.tex}.

OFS offers testing macro as well, whether the control sequence corresponds to the character, that is available in the font or not:

\knownchar \langle \text{character or accent+character} \rangle ? \iftrue \text{character is available} \else \text{character is unavailable or undefined} \fi
% example:
def\tryeuro{\knownchar \euro? \iftrue \euro \else Euro\fi}

### 3.6. The auxiliary macros for accents and characters

You can declare the default accents in OFS not only by the \accent primitive but by the macros \accentabove and \accentbelow too. The syntax follows:

\accentabove \langle \text{accent char} \rangle \langle \text{vertical skip} \rangle \langle \text{base char} \rangle
\accentbelow \langle \text{accent char} \rangle \langle \text{vertical skip} \rangle \langle \text{base char} \rangle

The \accentabove command put the \langle \text{accent char} \rangle above the \langle \text{base char} \rangle with the \langle \text{vertical skip} \rangle between them. The \accentbelow command does the same work, but put the \langle \text{accent char} \rangle below the \langle \text{base char} \rangle. In both cases characters are placed on the joint vertical axis and if the font is slanted then this axis is slanted too. The width of the resulting character is derived from the width of the \langle \text{base char} \rangle only. The macros are implemented by the \texttt{vbox}, \texttt{vtop} and \texttt{halign} primitives with the calculation of the (possibly) slanted axis.

The accented characters for accent above are commonly designed in the height 1 ex for most fonts. It means, that placing such character by \accentabove command needs the -1ex compensation:

\accentabove \langle \text{accent char} \rangle \langle -1ex \rangle \langle \text{base char} \rangle

In such case, the \accentabove has the same behavior as the \accent primitive. The difference is that you can compose more than one accent by \accentabove and \accentbelow macros. You can try:

\texttt{\textit{\accentabove {.}{.1ex}{\accentabove {,}{.1ex}{\texttt{\v A}}}}}

\texttt{PLAINTEX:} Unfortunately, the declaration macro \accentdef is not able to declare a macros which construct more than two accents. Moreover, the first accent has to be to join to the \langle \text{base char} \rangle as one compact character in the font. If you needs more accents then you can use the macros \accentabove and \accentbelow directly in the document.

\texttt{PLAINTEX:} Since the version OFS Feb. 2004, the macro \texttt{ofshexbox} is available. It acts very similar to the plain one \texttt{mathhexbox}, furthermore it can set the font in accordance to actual version. You can declare the family of four metrics by means of the command \texttt{ofshexboxdef}:

20
\ofshexboxdef \{family\}{(metrics-rm)}{(metrics-bf)}{(metrics-it)}{(metrics-bi)}
% example:
\ofshexboxdef 2 \{cmy\}{cmbsy10}{cmy\}{cmbsy10} % an example

The command \ofshexbox (family)(hexa-code) prints out the requested character. Its font is one of four declared ones and its size is defined by the command \fosize. The font choice depends on the actual version. If the version is different than \bf, \it, \bi, the \{metrics-rm\} is used.

OFS declares only (family) = 2 by default, because plain uses only \mathbox2...
The purpose of this macro was to define characters \S, \dag, \ddag, \P for CM-fonts/CSfonts. The way of the definition has to be independent on actual setting of mathematical fonts, but dependent on actual size and version. See the file ofs-8z.tex.

We can easily define the \euro symbol by means of \ofshexbox for every font encoding, where it is unavailable:

\ofshexboxdef \{TS1\}{tcrm1000}{tcx1000}{tcti1000}{tcbi1000}
\characterdef \euro * \{ofshexbox\{TS1\}BF

3.7. PLAINTEX: The fonts in mathematics (the second appearance)

The user interface to the math fonts was described in section 2.7. Now, it is the time to describe the principles of math fonts in detail.

The \setmath command calculates the text, index and indexindex sizes from its parameters. The results are stored into macros \textfosize, \scriptfosize and \scriptscriptfosize. The values are in the form at (dimen) or scaled (number) depending on the format of the parameters. Then the \setmath runs the macro \mathfonts. You can define the math fonts loading here but some conventions are recommended, see below. If the macro \setmath is run for the first time or the value \fomenc has been changed, then the \setmath runs the macro \mathchars. You can define the math encoding by the \matchode, \mathchardef etc. primitives here but some conventions are recommended, see below. The OFS serves the default values of the \mathfonts and \mathchars macros, see below.

You can load a whole math family (text, index and indexindex size of one font) in \mathfonts macro by the \loadmathfam command. This command has the following syntax possibilities:

%% % font is declared by::
\loadmathfam \{family\}[\{metrics\}] % metrics
\loadmathfam \{family\}[-\{version\}] % actual family version
\loadmathfam \{family\}[\{switch\}] % textual font switch
\loadmathfam \{family\}[X\{switch\}] % extending switch metrics

Examples:

\loadmathfam 0[tenrm/] % metrics is in accordance to the switch \tenrm
\loadmathfam 5[-bi/] % actual text family metrics, version bi
\newmathfam \symbfam
\loadmathfam \symbfam [\{psyr\}] % psyr metrics
\newmathfam \extitfam
\loadmathfam \extitfam [\textit{tenit/}] % extending metrics for tenit switch

This example shows, that the textual font with the \tenrm font switch is used for math family 0. In the family 5, fonts are initiated just like in the case of \setfonts [-bi/].
A new family \texttt{symbfam} is declared as well. Fonts of the metric \texttt{psyr} are initiated into it.

There exist a slight difference between usage of \texttt{\loadmathfam 5[tenbi/]} and \texttt{\loadmathfam 5[-bi/]} command. In the first case, OFS finds out a metrics of the \texttt{tenbi} switch and the same metrics is then used for all font sizes. The only modification is done by the key word \texttt{at(dimen)}. The other case means, that different sizes can use different metrics, but only if such a font ability is declared (see section 3.8).

The new family was in the example declared by means of the \texttt{\newmathfam}. It is an \texttt{\newmathfam} alternative command to \texttt{\newfam}. The reason for such a solution is evident. The plain macro \texttt{\newfam} is defined as \texttt{\outer} one. It means that it is not possible to use it inside any definition. Moreover the macro \texttt{\newmathfam} is local, so it spares some place for user families, than plain one \texttt{\newfam}. New mathematical families are in the basic mathematical encodings defined exactly by \texttt{\chardef}. The command \texttt{\lastfam=(number)} sets the maximal used value. Such a structure guarantees, that user can use \texttt{\newmathfam} later on and the new family numbers are allocated with numbers greater than \texttt{\lastfam}.

Let’s check out the principle of the \texttt{\loadmathfam} macro activity. This macro finds out a metrics, that corresponds to a given parameter. Next, the primitive \texttt{\font} is executed three times:

\begin{verbatim}
\font \langle name\rangle-Mt = \langle metric\rangle \texttt{textfosize}
\font \langle name\rangle-Ms = \langle metric\rangle \texttt{scriptfoshize}
\font \langle name\rangle-Mss = \langle metric\rangle \texttt{scriptscriptfoshize}
\textfont \langle math fam\rangle = \langle name\rangle-Mt
\scriptfont \langle math fam\rangle = \langle name\rangle-Ms
\scriptscriptfont \langle math fam\rangle = \langle name\rangle-Mss
\end{verbatim}

Nevertheless \texttt{name} is the text of the parameter \texttt{\loadmathfam}, that declares metrics (switch, version or metrics).

The \texttt{name} is generated as name of the \texttt{\fontswitch} or the name of the \texttt{\metric} if only the \texttt{\metrics} is given as parameter of the \texttt{\loadmathfam}.

The fonts of math family 3 are loaded without of the size changes of index and indexindex fonts in plain\TeX. If you need this feature then you can use the prefix \texttt{\noindexsize} before \texttt{\loadmathfam}. The macro \texttt{\loadmathfam} loads all three fonts at the same \texttt{\textfoshize} size. Example:

\begin{verbatim}
\noindexsize\loadmathfam 3[tenex/]% Standard extra symbols from CM
\end{verbatim}

OFS defines four different macros for math font loading. Look at \texttt{ofsdef.tex} file for these definitions. Which of these four macros is used depends on the contents of macros \texttt{\fomenc} and \texttt{\mathversion}. We assume two possibilities of \texttt{\fomenc}: CM or PS and two possibilities of \texttt{\mathversion}: normal and bold. OFS defines two macros with mathcodes. Which macro is used depends on the contents of the macro \texttt{\fomenc}. The list of these macros follows:

\begin{verbatim}
\loadCMnormalmath — loads CM fonts in “normal” version.
\loadCMboldmath — loads CM fonts in “bold” version.
\loadPSnormalmath — loads PostScript fonts in “normal” version.
\loadPSboldmath — loads PostScript fonts in “bold” version.
\setCMmathchars — keeps the mathcodes from plain.
\setPSmathchars — sets the mathcodes in order to some characters are used from Symbol font.
\end{verbatim}
OFS sets the following defaults in the ofsdef.tex file:

\ifx \fomenc\undefined \def\fomenc{PS}\fi
\def\mathversion{normal}
\def\defaultmathfonts{\csname load\fomenc\mathversion math\endcsname}
\defaultmathfonts
\def\defaultmathchars{\csname set\fomenc mathchars\endcsname}
\defaultmathchars
\def\mathfonts{\defaultmathfonts}
\def\mathchars{\defaultmathchars}

It is possible to add the another math families to the list of loaded math families in \loadmath and \mathchars macros. You can do it, for example, by the following code:

The example how to add the Euler fraktur from AMS follows:

\input amsfn % here are declared metrics eufm and eufb
\addcmd\mathfonts{\def\tmpa{bold}\
  \ifx\mathversion\tmpa \def\tmpa{b}\else\def\tmpa{b}\fi
  \newmathfam\frakfam \loadmathfam\frakfam [/euf\tmpa]}
\def\frak#1{{\fam\frakfam#1}}

Another examples of mathematical families declaration are located in files amsfn.tex, txfn.tex and mtfn.tex. These files define the implicit groups of the mathematical fonts for AMS, TX, PX, MT encodings. Finally they also contain comments (including examples), how to load additional font families by means of the command \addcmd.

I would like to load all OFS font declarations in the ini\TeX (see the Ok\TeX project), but I want to spare the \TeX memory as much as possible too. So I suggest not to load the large definitions containing declarations of mathematical fonts encodings by means of \mathchardef, etc. immediately, but during the first use in the document. So the OFS version Apr. 2004 contains redefined macro \setPSmathchars:

\def\setPSmathchars{\setPSmathchars ofs-ps;}

\mathencread The command \mathencread \{file\}; loads the encoding commands included in the file \{file.tex\}. The file ofs-ps.tex contains encoding commands for PS encoding, ofs-ams.tex contains encoding commands for AMS encoding, etc.

The files ofs-ps.tex, ofs-tx.tex etc. contain encoding commands “packed” into groups and defined by the \mathencdef command. By default, this command acts this way: “define, run and forget”. Such a model spares the memory, but a disadvantage of its procedure is, that during repeated changes, encoding files are read over and over again. Mostly it does not matter at all, because the mathematical encoding is the same for the whole document. If this model is for somebody not suitable, the solution can be found in the macros \mathencread and \mathencdef. They can be redefined by the way, that the files are read only once and during new execution of \mathchars, encoding commands are read out of remembered macros.

The command \mathencread \{file\}; works in group. The catcodes are set in accordance to plain\TeX (\texttt{\catcode'@=11} excluded) and the \{file\}.tex is read. The \globaldefs=1 are during the reading, the empty lines and ends of lines are ignored. The commands \mathencdef run and forget defined macros after group enclosure. It means, that these macros run the \mathchardef commands in the sense of local settings.

Let us explain an ensurance of restoring the default values during switching between mathematical encodings. The command \setmath runs the macro \mathcharsback in a time instant before \mathchars. It serves the restoring of the mathematical encoding to the default state in accordance to plain\TeX. This macro is set to \relax by default,
because the mathematical encoding is set in accordance to plain\TeX. If the command \texttt{\set⟨fomenc⟩\mathchars} changes the values preset in the plain\TeX, it should also define the \texttt{\mathcharsback} macro. A procedure of setting the values into the initial state have to be stored in it. A command \texttt{\mathencreads-cm;} can be used in the macro \texttt{\mathcharsback}, because the file \texttt{ofs-cm.tex} contains declarations of the mathematical encoding in accordance to plain\TeX.

If we insist on declaring our own mathematical encodings (different from preprepared PS, CM, AMS etc.), the next principle has to be fulfilled: in all versions of mathematical fonts (normal/bold/etc.) should be set the numbers of the mathematical families by the same way and ended by the same \texttt{\lastfam}. Such a principle is necessary, because after the version switching, the macro \texttt{\mathchars} is not run again. It is not recommended to change especially the family numbers, that are linked with the macro \texttt{\set⟨encoding⟩\mathchars}.

Let us now describe another macros, that helps us with the mathematical encoding declaration. Macro \texttt{\hex} converts a number to a singledigit hexadecimal number. The usage of such a macro can be found in the files \texttt{ofs-ps.tex}, \texttt{ofs-tx.tex}, etc.

You can define the control sequences which are working in both: text mode and math mode. You can use the \texttt{\safemathchardef} macro instead \texttt{\mathchardef} primitive in \texttt{\mathchars} macro for this purpose:

\begin{verbatim}
\safemathchardef \langle sequence \rangle \langle number \rangle
\end{verbatim}

If the \texttt{\langle sequence \rangle} is not defined then \texttt{\safemathchardef} does the same work as \texttt{\mathchardef} primitive. If the \texttt{\langle sequence \rangle} is defined (we assume that this definition is used in text mode) then \texttt{\safemathchardef} saves the meaning of \texttt{\langle sequence \rangle} in \texttt{\T\langle sequence \rangle = \langle number \rangle} and it defines \texttt{\langle sequence \rangle} by the following way:

\begin{verbatim}
\def \langle sequence \rangle {
\ifmmode \expandafter \M \langle sequence \rangle \else \expandafter \T \langle sequence \rangle \fi}
\end{verbatim}

Now, the \texttt{\langle sequence \rangle} works in both: text and math mode. If the \texttt{\safemathchardef} is applied on the same \texttt{\langle sequence \rangle} repetitively, then the second and another use of \texttt{\safemathchardef} does nothing.

We assume that all declarations of characters in text mode are performed before the first use of \texttt{\setmath} and that the \texttt{\safemathchardef} macros are used in \texttt{\mathchars} macro.

The \texttt{\safemathaccentdef} performs similar as \texttt{\safemathchardef}. Only the \texttt{\mathaccentdef} macro is used instead \texttt{\mathchardef} primitive. This macro does roughly the following:

\begin{verbatim}
\def \langle sequence \rangle {\langle mathaccent \langle number \rangle \rangle}
\end{verbatim}

If you do not want to declare the whole new math family for only one character or a few characters (the number of math families is restricted to 16 for one formula in \TeX) then you can use the \texttt{\pickmathfont} macro. This macro has the following syntax:

\begin{verbatim}
\pickmathfont \langle metric \rangle \langle text \rangle
\end{verbatim}

\texttt{\% example: \mathbin \{\pickmathfont \{psybo\}\{\char"C4\}\}}

The \texttt{\pickmathfont} command uses the \texttt{\font} primitive with the given \texttt{(metric)} and it prints the \texttt{(text)} in this font. The result is an atom of type Ord in the math list. The
appropriate size is set (text/index/indexindex) because the \mathchoice primitive is
used by \pickmathfont macro and the \font is loaded three times for each size.

3.8. The different metrics for different sizes of font

\textsc{LaTeX}: This problem is solved in NFSS, see the syntax of the \texttt{fd} files in NFSS
documentation. OFS for \LaTeX\ does no more things.

\textsc{PlainTeX} (to the end of this section): There exist special font families (Computer Modern,
for instance) where the different metrics are used for the different font sizes. This feature
is implemented in OFS too. The \langle \text{metric} \rangle parameter of the commands \loadtextfam,
\loadmathfam and \pickmathfont can be only the \langle \text{symbolic name} \rangle and not the name
of \texttt{tfm} file exactly. In such case, the \langle \text{real metric} \rangle is calculated from the font size and
from the data stored by a set of \registertfm commands. The \registertfm has a
following syntax:

\begin{verbatim}
\registertfm \langle symbolic name \rangle \langle space \rangle \langle from \rangle-\langle to \rangle \langle space \rangle \langle real metric \rangle \langle space \rangle
\end{verbatim}

The \langle from \rangle and \langle to \rangle parameters have to include the unit and they declares the
interval of sizes with the following property: if the desired size of font is in interval
\langle from \rangle-\langle to \rangle then the \langle real metric \rangle is used instead the \langle symbolic name \rangle. You need to
use more \registertfm commands to the same \langle symbolic name \rangle in order to use differ-
ent \langle real metric \rangle for different \langle from \rangle-\langle to \rangle intervals. The \langle from \rangle-\langle to \rangle interval is closed
(interval including the boundary values) but the next \registertfm has a precedence
over previous one. Then you can construct the non closed intervals by the choose of the
right order of \registertfm commands. See the \texttt{ofsdef.tex} file for an example.

If the both parameters \langle from \rangle and \langle to \rangle are empty then the \langle real metric \rangle is used
when the \texttt{scaled} keyword is given in desired font size or if desired font size does not
lie in any declared interval. You can use the \texttt{*} symbol in \langle to \rangle parameter—it means the
infinity.

The command \registertfm \langle name \rangle \langle space \rangle - \langle space \rangle - \langle space \rangle erases the pre-
vious registrations for a given \langle name \rangle. Moreover, it marks \langle name \rangle as an unavailable
font. OFS then acts, as if the corresponding variant has not been declared at all. It
gives us a possibility to mark nonexistent variants of a declared family, but only for
certain encodings (see \texttt{cmssbxti} in the file \texttt{ofsdef.tex}).

The parameters \langle symbolic name \rangle and \langle real metric \rangle are expanded during the command \registertfm is working. Thus the \texttt{fotenc} macro should not be used in these
parameters. But you can use an \texttt{edef} construction if necessary.

The macro \ registertfont is an abbreviation of the repeated execution of the command \registertfm to all EC fonts sizes between 0500 up to 3583. The macro \registertfont is an abbreviation for sizes 0800 do 3583, that are used for the type-
writer fonts. Definitions and usage of these macros are located in the file \texttt{ofsdef.tex}. Macros can be used for another fonts as well. But they have to have similarly scaled sizes
like EC fonts (LH fonts with Russian alphabet or the fonts derived from CM super).

3.9. The limitations of usage the font family for certain encodings

\textsc{LaTeX}: The usage of a family in a given encoding is dependent on the existence of \texttt{fd}
file. It is all controlled by NFSS.

\textsc{PlainTeX} (to the end of this section): The OFS version Feb. 2004 introduces a command
\registeenc. This command limits the usage of a declared font family only for a
certain encoding. If the user writes \texttt{setfonts} and requires the family in an unregistered
encoding, the OFS prints out a warning on the terminal and does not switch to the requested family. It avoids an attempt to loading a nonexisting font metrics. The command \registerenc has two parameters.

\registerenc \langle FamilyName \rangle: \langle encoding \rangle \langle space \rangle
\registerenc Times: 8t \ % example
\registerenc Times: 8z \ % Times is registered for two encodings

If the family is not registered for any encoding, OFS then suggests, that it is available in every encoding (dingbats fonts for example).

The family usage is limited for a certain encoding by \registerenc in declaration files. The user can add another encoding by the command \registerenc. The command \registerenc (FamilyName): * means, that the family is available for arbitrary encodings.

The parameter \langle FamilyName \rangle can remain empty. In such case, the \registerenc uses the family declared in the last command \ofsdeclarefamily.

It is possible to find out, whether the family is registered for actual value of the encoding:\n
\registeredfam \fotenc:
\registeredfam \langle Family Name \rangle? \iftrue
  Family has the \fotenc registered or any encoding is enabled
  or is not declared at all.
\else
  Family has registered the encoding,
  but \fotenc is not among them
\fi

If \setfonts[(FamilyName)/] is run for an undeclared family or for a family with unavailable encoding then OFS prints out a warning on the terminal and returns the \setfontsOK with the value \undefined. The \setfontsOK has assigned the value \relax, whenever the font is successfully set.

4. The license

The OFS package may be used by everybody without any license fees. Everybody can re-distribute this package, if no changes in files readme.ofs, ofs.tex, ofsddef.tex, ofs.sty, ofs-8z.tex, ofs-8t.tex, a35.tex, a35.sty, ofsdoc.tex, ofsdoc-e.tex, ofsmtdef.tex are done and all these files are included in the re-distribution. Only the author has a right to change these files and to change version of this package. If you need to change the content of any file mentioned above, you have to rename it. This package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

Prague 08/16/2001 the author: Petr Olšák
5. History

8/16/2001 — the first version introduced
10/24/2002 — some petty changes realized in documentation, considering the changes in the version OFS Oct. 2002. The command \addcmd has been added. The optional parameters \(<Variant>\) in the command \loadtextfam were enabled.

12/29/2002: the English documentation in the file ofsdoc-e.tex is written. This is more or less the translation of the Czech original documentation from the file ofsdoc.tex. Sorry for my poor English.

I want to say a word of thanks to Matěj Cepl (www.ceplovi.cz) who has made a proofreading of this English version.

2/10/2004 — All modifications consider only OFS for plain:
+ Handling the macro declarations dependent on encoding has been improved as well as the exception declarations and registering of the encoding for a chosen family.
  See section 3.5 and section 3.9.
+ Added the possibility to redefine the family (by the additional loading the declaring file, that modifies the properties of the default families).
+ Declaration of CMRoman, CMSans and CMTypewriter for 8t encoding by means of EC fonts.
+ Added the support for extended encoding 8c.
+ Modified the declaration of the mathematical encoding called PS. \int, \sum and \prod produces greater characters in the display mode.
  + Defined a macro ofshbbox and ofshbboxdef.
+ A new version of the interactive macro ofstest.tex.
+ Introduced the directory \examples/. There will be continuously added the examples of OFS usage there.

3/12/2004 — OFS for plain:
+ Encoding files are read directly from the \loadtextfam. That means the possibility to place the mapping metrics into the encoding files by means of the command \registerenc (I will use this ability for LANG).
  + \characterdef respects predefined macro an does not redefines it.
+ \showfonts reimplemented. It spares the memory and time, when operates with the large font lists. Warning: if you have used macros, that plays upon the internal macro \listfamilies, this will not operate any more. Since this version is not the obsolete ofscatal.tex functional too. Instead of it is possible to use ofstest.tex.
+ ofstest.tex modified. It operates with the newly imlemented list of families \ofslistfamilies.

4/2/2004 — \plaincatcodes added before reading the files ofs-\langle encoding\rangle .tex.
+ \safelet and \protectreading introduced.
+ The space behind the \langle character \rangle in the \accentdef is optional.
+ Added the option \loadmathfam \langle family\rangle[\langle version\rangle]/. The declaration for mathematical encodings CM, PS, AMS, TX, PX, MT has been extended and remade.
+ English documentation upgraded. Thaks to Tomáš Komárek.
6. Index of macros defined in OFS

This index includes only the pointers to the pages where the macro is introduced (not only mentioned). The macro name is written in the margin on that place. The index is generated after first run of TEX.

The internal or auxiliary macros in \texttt{ofs.tex} are listed here too but they are not mentioned in the text, thus they have no pointer to the page in this index. Only a short comment is appended here. The version of OFS (\texttt{PLAIN} or \texttt{LATEX}) is listed near the each macro name in this index.

\begin{verbatim}
\accentabove (PLAIN) 20
\accentbelow (PLAIN) 20
\accentdef (PLAIN) 17
\accentdel (PLAIN) 18
\accentdefori, \accentdelori (PLAIN) internal, stores the default macro purport
\accentnodef (PLAIN) internal, \texttt{\\def\{macro\}\{\printaccent{\{macro\}\{#1\}}}\}
\addcmd (PLAIN+LATEX) 6
\bif (PLAIN+LATEX) 2
\bi (PLAIN+LATEX) 2
\bifam (PLAIN) internal, math family for BoldItalic
\calculatemetricfile (PLAIN) internal, defines \texttt{\metricfile} by fontsize
\catcodesloop (PLAIN) internal, sets the character categ. according to a given num.
\characterdef (PLAIN) 15
\characterdel (PLAIN) 18
\characterdefori, \characterdelori (PLAIN) internal, stores default macro purport
\characternodef (PLAIN) internal, \texttt{\\def\{macro\}\{\printcharacter{\{macro\}}}\}
\currentfamily (PLAIN) 15
\currentfomenc (PLAIN) internal, the name of last used math. encoding
\currentvariant (PLAIN) 14
\declaredfamily (PLAIN) internal, contains the name of the last declared family
\defaultextraenc (PLAIN) 13
\defaultmathchars (PLAIN) 23
\defaultmathfonts (PLAIN) 23
\defpttotmpa (PLAIN) internal, does \texttt{\def\tmpa{pt}}, if the unit is omitted
\detailfontmessages (PLAIN) 10
\displayfontmessages (PLAIN) 10
\displaymessage (PLAIN) internal 10
\doclassdef (PLAIN) internal, for use of \texttt{\characterdef} macro
\doextchar (PLAIN) internal, for use of \texttt{\extchar} macro
\dosaframathdef (PLAIN) internal, for use of \texttt{\safemath*def} macros
\donumbercharacterdef (PLAIN) internal, for use of \texttt{\characterdef} macro
\endOFSmacro (PLAIN) internal, the reading of \texttt{[\{file\}, ...]} after \texttt{\input \ofs}
\expandaction (PLAIN) 11
\extchar (PLAIN+LATEX) 15
\extraenc (PLAIN) 13
\extrafont (PLAIN) 15
\fontdef (PLAIN+LATEX) 5
\fontloadmessage (PLAIN) internal, the tracing of \texttt{\font} primitives
\fontmessage (PLAIN) internal, three values: nothing, \texttt{\wlog} or \texttt{\displaymessage}
\fontprefix (PLAIN) internal, two values: nothing or \texttt{\global}
\fontusage (PLAIN+LATEX) 3
\forsize (PLAIN) 12
\fomenc (PLAIN) 9, 9
\fotenc (PLAIN) 8, 12
\fragilecommand (PLAIN) 11
\fragilecommand! (PLAIN) 11
\end{verbatim}

28
\texttt{\textbackslash{}hex} (PLAIN) 24
\texttt{\textbackslash{}ifknownfam} (PLAIN+LATEX) 7
\texttt{\textbackslash{}isunitpresent} (PLAIN) internal, solves the case of empty unit
\texttt{\textbackslash{}it} (PLAIN+LATEX) 2
\texttt{\textbackslash{}knownchar} pl 20
\texttt{\textbackslash{}knownfam} pl 7
\texttt{\textbackslash{}lastfam} pl 22
\texttt{\textbackslash{}lccodes, \textbackslash{}lccodesloop} (PLAIN) 19
\texttt{\textbackslash{}loadCMboldmath} (PLAIN) 22
\texttt{\textbackslash{}loadCMnormalmath} (PLAIN) 22
\texttt{\textbackslash{}loadingenc} (PLAIN) 8, 14, 18
\texttt{\textbackslash{}loadmathfam} (PLAIN) 21
\texttt{\textbackslash{}loadPSboldmath} (PLAIN) 22
\texttt{\textbackslash{}loadPSnormalmath} (PLAIN) 22
\texttt{\textbackslash{}loadtextfam} (PLAIN) 12, 13
\texttt{\textbackslash{}logfontmessages} (PLAIN) 10
\texttt{\textbackslash{}mathaccentdef} (PLAIN) 24
\texttt{\textbackslash{}mathchars} (PLAIN) 21
\texttt{\textbackslash{}mathcharsback} (PLAIN) 23
\texttt{\textbackslash{}mathencdef} (PLAIN) 23
\texttt{\textbackslash{}mathencread} (PLAIN) 23
\texttt{\textbackslash{}mathfonts} (PLAIN) 21
\texttt{\textbackslash{}mathversion} (PLAIN) 9
\texttt{\textbackslash{}metricfile} (PLAIN) internal, the metric name if \texttt{\textbackslash{}font} primitive is used
\texttt{\textbackslash{}metrictime} (PLAIN) internal, expands to the metric of \texttt{\textbackslash{}tmpa} font
\texttt{\textbackslash{}modifydef} (PLAIN) 18
\texttt{\textbackslash{}modifyenc} (PLAIN) 12, 18
\texttt{\textbackslash{}modifylist} (PLAIN) 19
\texttt{\textbackslash{}modifyread} (PLAIN) 19
\texttt{\textbackslash{}newfamily} (PLAIN) auxiliary, the given family name
\texttt{\textbackslash{}newmathfam} (PLAIN) 22
\texttt{\textbackslash{}newmodifylist} (PLAIN) 18
\texttt{\textbackslash{}newvariant} (PLAIN) 12, 14
\texttt{\textbackslash{}nofontmessages} (PLAIN) 10
\texttt{\textbackslash{}noindexsize} (PLAIN) 22
\texttt{\textbackslash{}noPT} (PLAIN) internal, removes \texttt{pt} from \texttt{\the\{dimen\}} [TBN, pg. 80]
\texttt{\textbackslash{}ofssaddenticolist} (PLAIN) internal, adds parameter into the list \texttt{\textbackslash{}newmodifylist}
\texttt{\textbackslash{}OFSdeclarefamily} (LATEX) 12
\texttt{\textbackslash{}OFSdeclarefam} (PLAIN) 12
\texttt{\textbackslash{}OFSextraencoding} (LATEX) 12
\texttt{\textbackslash{}OFSfamily} (LATEX) 7
\texttt{\textbackslash{}OFSfamilydefault} (LATEX) 7
\texttt{\textbackslash{}ofshexbox} (PLAIN) 20
\texttt{\textbackslash{}ofshexboxdef} (PLAIN) 20
\texttt{\textbackslash{}ofssinput} (PLAIN) internal, reads file with \texttt{\textbackslash{}globaldefs=1}, ignores endlinechars
\texttt{\textbackslash{}ofsslistfamilies} (PLAIN) internal, family list for \texttt{\textbackslash{}showfonts}
\texttt{\textbackslash{}ofsslistfamily} (PLAIN) internal, invokes the family in \texttt{\textbackslash{}listfamilies}
\texttt{\textbackslash{}ofsslistvariants} (PLAIN) internal, text for listing of variants to the log
\texttt{\textbackslash{}ofsslisttext} (PLAIN) internal, invokes the text in \texttt{\textbackslash{}listfamilies}
\texttt{\textbackslash{}ofssloadfont} (PLAIN) internal, loads one font
\texttt{\textbackslash{}ofssloadfontori} (PLAIN) internal, loads one font
\texttt{\textbackslash{}ofssmeaning} (PLAIN) internal, removes the word \texttt{letter/character} from \texttt{\meaning}
\texttt{\textbackslash{}ofssmessageheader} (PLAIN) internal, header of the messages
\texttt{\textbackslash{}OFSnormalvariants} (LATEX) 12
\texttt{\textbackslash{}OFSprocessoptions} (LATEX) 11
\texttt{\textbackslash{}OFSputfamlist} (LATEX) 12
\texttt{\textbackslash{}ofsputfamlist} (PLAIN) 12
\ofsremovefromlist (PLAIN) internal, erases the family from the list
\OFSversion (PLAIN+LATEX) internal, date and version of OFS
\orifosize (PLAIN) auxiliary, saves \fsize value
\origTeX (PLAIN) internal, original definition of \TeX logo
\oriloadfam (PLAIN) auxiliary, saves \loadtextfam value
\pickmathfont (PLAIN) 24
\plaincatcodes (PLAIN) internal, sets the chars categories according to plain\TeX
\printaccent (PLAIN) 17
\printaccentwarn (PLAIN) 17
\printcharacter (PLAIN) 16
\printcharacterwarn (PLAIN) 16
\processOFSoption (PLAIN) internal, for use of \endOFSmacro
\protectreading (PLAIN) 12
\readfamvariant (PLAIN) internal, checks if the variant is given
\readfirsttoken (PLAIN) internal, returns the first token of text separated by :\end
\readfosize (PLAIN) internal, inserts the \fsize value to \dimen0
\readmag (PLAIN) internal, calculates \fsize if \mag<decimal number> is given
\readOFSoptions (PLAIN) internal, for use of \endOFSmacro
\readothertokens (PLAIN) internal, returns the second and other tokens to :\end
\readsixdigits (PLAIN) internal, for rounding algorithm
\registeredfam (PLAIN) 26
\registerECfont (PLAIN) 25
\registerECTTfont (PLAIN) 25
\registerenc (PLAIN) 13, 25
\registertfm (PLAIN) 13, 25
\restorefontid (PLAIN) internal, restores font name, see \savefontid
\rm (PLAIN+LATEX) 2
\runmodifylist (PLAIN) 15, 19
\safelet (PLAIN) 17
\safeletwarn (PLAIN) 17
\safemathaccentdef (PLAIN) 24
\safemathchardef (PLAIN) 24
\savefontid (PLAIN) internal, saves font name for Overfull messages
\savetokenname (PLAIN) internal, similar to \string without backslash
\scriptfsize (PLAIN) 21
\scriptsixdigits (PLAIN) 21
\separeofsvariant (PLAIN) internal, separates the \langle Variant \rangle in \loadtextfam
\setextrafont (PLAIN+LATEX) 15
\setCMmathchars (PLAIN+LATEX) 22
\setfonts (PLAIN+LATEX) 4, 14, 15
\setfontshook (PLAIN) 13
\setsixdigits (PLAIN) 26
\setfontfamily (PLAIN) internal, \setfonts, if the variant is given
\setfsize (PLAIN) internal, calculate the value of \fsize
\setmath (PLAIN) 8, 21
\setPSmathchars (PLAIN) 22
\setsimplemath (PLAIN) 8
\setsinglefont (PLAIN) internal, \setfonts, if the variant is given
\setsinglefontname (PLAIN) internal, removes the possible \at\langle dimen \rangle from \langle metric \rangle
\sgfamily (PLAIN) internal, the information about the family name
\sgvariant (PLAIN) internal, the information about the variant name
\showfonts (PLAIN+LATEX) 3
\singlefont (PLAIN) internal
\singlefontname (PLAIN) internal, removes the \at\langle dimen \rangle from metric name
\skipfirststep (PLAIN) 19
\slantcorrrection (PLAIN) internal, for use of \accentabove and \accentbelow
\storeofsvariant (PLAIN) internal, separates optional parameter of \loadtextfam
\textbf{7. References}