pLaTeX 2ε のフォントコマンド v1.3n

Ken Nakano & Hideaki Togashi

作成日：2004/08/10

Contents

1 概要

ここでは、和文書体を NFSS2 のインターフェイスで選択するためのコマンドやマクロについて説明をしています。また、フォント定義ファイルや初期設定ファイルなどの説明もしています。新しいフォント選択コマンドの使い方については、`fntguide.tex` や `usrguide.tex` を参照してください。

第 1 節 この節です。このファイルの概要と DOCSTRIP プログラムのためのオプションを示しています。

第 2 節 実際のコードの部分です。

第 3 節 プリロードフォントやエラーフォントなどの初期設定について説明をしています。

第 4 節 フォント定義ファイルについて説明をしています。

1.1 DOCSTRIP プログラムのためのオプション

DOCSTRIP プログラムのためのオプションを次に示します。

オプション	意味
plcore	`plfonts.ltx` を生成します。
trace	`ptrace.sty` を生成します。
JY1mc	横組用、明朝体のフォント定義ファイルを生成します。
JY1gt	横組用、ゴシック体のフォント定義ファイルを生成します。
JT1mc	縦組用、明朝体のフォント定義ファイルを生成します。
JT1gt	縦組用、ゴシック体のフォント定義ファイルを生成します。
pldefs	`pldefs.ltx` を生成します。次の 4 つのオプションを付加することで、プリロードするフォントを選択することができます。デフォルトは 10pt です。
xpt	10pt プリロード
xipt	11pt プリロード
xiipt	12pt プリロード
ori	`plfonts.tex` に似たプリロード

2 コード

この節で、具体的に NFSS2 を拡張するコマンドやマクロの定義を行なっています。

2.1 準備

NFSS2 を拡張するための準備です。和文フォントの属性を格納するオブジェクトや長さ変数、属性を切替える際の判断材料として使うリストなどを定義しています。

2.1.1 和文フォント属性

ここでは、和文フォントの属性を格納するためのオブジェクトについて説明をしています。

`\k@encoding` `\ck@encoding` `\cy@encoding` `\ct@encoding`

和文エンコードを示すオブジェクトです。`\ck@encoding` は、最後に選択された和文エンコード名を示しています。`\cy@encoding` と `\ct@encoding` はそれぞれ、最後に選択された、横組用と縦組用の和文エンコード名を示しています。

```
1 ⟨*plcore⟩
```

```
\let\k@encoding\@empty
\let\ck@encoding\@empty
\def\cy@encoding{JY1}
\def\ct@encoding{JT1}
```

\k@family 和文書体のファミリを示すオブジェクトです。

```
\let\k@family\@empty
```

\k@series 和文書体のシリーズを示すオブジェクトです。

```
\let\k@series\@empty
```

\k@shape 和文書体のシェイプを示すオブジェクトです。

```
\let\k@shape\@empty
```

\curr@kfontshape 現在の和文フォント名を示すオブジェクトです。

```
\def\curr@kfontshape{\k@encoding/\k@family/\k@series/\k@shape}
```

\rel@fontshape 関連付けされたフォント名を示すオブジェクトです。

```
\def\rel@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

2.1.2 長さ変数

ここでは、和文フォントの幅や高さなどを格納する変数について説明をしています。

頭文字が大文字の変数は、ノーマルサイズの書体の大きさで、基準値となります。これらは、jart10.clo などの補助クラスファイルで設定されます。

小文字だけからなる変数は、フォントが変更されたときに（\selectfont 内で）更新されます。

\Cht \Cht は基準となる和文フォントの文字の高さを示します。\cht は現在の和文フォント\cht トの文字の高さを示します。なお、この “高さ” はベースラインより上の長さです。

```
\newdimen\Cht
\newdimen\cht
```

\Cdp \Cdp は基準となる和文フォントの文字の深さを示します。\cdp は現在の和文フォント\cdp トの文字の深さを示します。なお、この “深さ” はベースラインより下の長さです。

```
\newdimen\Cdp
\newdimen\cdp
```

\Cwd \Cwd は基準となる和文フォントの文字の幅を示します。\cwd は現在の和文フォント\cwd トの文字の幅を示します。

```
\newdimen\Cwd
\newdimen\cwd
```

\Cvs \Cvs は基準となる行送りを示します。ノーマルサイズの\baselineskip と同値です。\cvs は現在の行送りを示します。

```
\newdimen\Cvs
\newdimen\cvs
```

\Chs \Chs は基準となる字送りを示します。\Cwd と同値です。\chs は現在の字送りを示
\chs します。

```
\newdimen\Chs
\newdimen\chs
```

\cHT \cHT は、現在のフォントの高さに深さを加えた長さを示します。\set@fontsize コマンド（実際は\size@update）で更新されます。

```
\newdimen\cHT
```

2.1.3 一時コマンド

\afont LaTeX 内部の\do@subst@correction マクロでは、\fontname\font で返される外部フォント名を用いて、LaTeX フォント名を定義しています。したがって、\font をそのまま使うと、和文フォント名に欧文の外部フォントが登録されたり、縦組フォント名に横組用の外部フォントが割り付けられたりしますので、\jfont か\tfont を用いるようにします。\afont は、\font コマンドの保存用です。

```
\let\afont\font
```

2.1.4 フォントリスト

ここでは、フォントのエンコードやファミリの名前を登録するリストについて説明をしています。

pLaTeX 2ε の NFSS2 では、一つのコマンドで和文か欧文のいずれか、あるいは両方を変更するため、コマンドに指定された引数が何を示すのかを判断しなくてはなりません。この判断材料として、リストを用います。

このときの具体的な判断手順については、エンコード選択コマンドやファミリ選択コマンドなどの定義を参照してください。

\inlist 次のコマンドは、エンコードやファミリのリスト内に第二引数で指定された文字列があるかどうかを調べるマクロです。

```
\def\inlist@#1#2{%
  \def\in@@##1<#1>##2##3\in@@{%
    \ifx\in@##2\in@false\else\in@true\fi}%
  \in@@#2<#1>\in@\in@@}
```

\enc@elt \enc@elt と\fam@elt は、登録されているエンコードに対して、なんらかの処理を
\fam@elt 逐次的に行ないたいときに使用することができます。

```
\def\fam@elt{\noexpand\fam@elt}
\def\enc@elt{\noexpand\enc@elt}
```

\fenc@list \fenc@list には、\DeclareFontEncoding コマンドで宣言されたエンコード名が
\kenc@list 格納されていきます。

\kyenc@list \kyenc@list には、\DeclareYokoKanjiEncoding コマンドで宣言されたエン
\ktenc@list コード名が格納されていきます。\ktenc@list には、\DeclareTateKanjiEncoding コマンドで宣言されたエンコード名が格納されていきます。

ここで、これらのリストに具体的な値を入れて初期化をするのは、リストにエンコードの登録をするように\DeclareFontEncodingを再定義する前に、欧文エンコードが宣言されるため、リストに登録されないからです。

```
\def\fenc@list{\enc@elt<OML>\enc@elt<T1>\enc@elt<OT1>\enc@elt<OMS>%
               \enc@elt<OMX>\enc@elt<TS1>\enc@elt<U>}
\let\kenc@list\@empty
\let\kyenc@list\@empty
\let\ktenc@list\@empty
```

\kfam@list
\ffam@list
\notkfam@list
\notffam@list

\kfam@listには、\DeclareKanjiFamilyコマンドで宣言されたファミリ名が格納されていきます。

\ffam@listには、\DeclareFontFamilyコマンドで宣言されたファミリ名が格納されていきます。

\notkfam@listには、和文ファミリではないと推測されたファミリ名が格納されていきます。このリストは\fontfamilyコマンドで作成されます。

\notffam@listには欧文ファミリではないと推測されたファミリ名が格納されていきます。このリストは\fontfamilyコマンドで作成されます。

ここで、これらのリストに具体的な値を入れて初期化をするのは、リストにファミリの登録をするように、\DeclareFontFamilyが再定義される前に、このコマンドが使用されるため、リストに登録されないからです。

```
\def\kfam@list{\fam@elt<mc>\fam@elt<gt>}
\def\ffam@list{\fam@elt<cmr>\fam@elt<cmss>\fam@elt<cmtt>%
               \fam@elt<cmm>\fam@elt<cmsy>\fam@elt<cmex>}
```

つぎの二つのリストの初期値として、上記の値を用います。これらのファミリ名は、和文でないこと、欧文でないことがはっきりしています。

```
\let\notkfam@list\ffam@list
\let\notffam@list\kfam@list
```

2.1.5 支柱

行間の調整などに用いる支柱です。支柱のもととなるボックスの大きさは、フォントサイズが変更されるたびに、\set@fontsizeコマンドによって変化します。

フォントサイズが変更されたときに、\set@fontsizeコマンドで更新されます。

\tstrutbox
\zstrutbox

\tstrutboxは高さと深さが5対5、\zstrutboxは高さと深さが7対3の支柱ボックスとなります。これらは縦組ボックスの行間の調整などに使います。なお、横組ボックス用の支柱は\strutboxで、高さと深さが7対3となっています。

```
\newbox\tstrutbox
\newbox\zstrutbox
```

\strut
\tstrut
\zstrut

\strutboxは\yoko ディレクションで組まれていますので、縦組ボックス内で\unhcopyをするとエラーとなります。このマクロはltplain.dtxで定義されています。

```
\def\strut{\relax
```

```
  \ifydir
    \ifmmode\copy\strutbox\else\unhcopy\strutbox\fi
  \else
    \ifmmode\copy\tstrutbox\else\unhcopy\tstrutbox\fi
  \fi}
\def\tstrut{\relax\hbox{\tate
   \ifmmode\copy\tstrutbox\else\unhcopy\tstrutbox\fi}}
\def\zstrut{\relax\hbox{\tate
   \ifmmode\copy\zstrutbox\else\unhcopy\zstrutbox\fi}}
```

2.2 コマンド

次のコマンドの定義をしています。

コマンド	意味
\Declare{Font|YokoKanji|TateKanji}Encoding	エンコードの宣言
\Declare{Yoko|Tate}KanjiEncodingDefaults	デフォルトの和文エンコードの宣言
\Declare{Font|Kanji}Family	ファミリの宣言
\DeclareKanjiSubstitution	和文の代用フォントの宣言
\DeclareErrorKanjiFont	和文のエラーフォントの宣言
\DeclareFixedFont	フォントの名前の宣言
\reDeclareMathAlphabet	和欧文を同時に切り替えるコマンド宣言
\{Declare|Set}RelationFont	従属書体の宣言
\userelfont	欧文書体を従属書体にする
\selectfont	フォントを切り替える
\set@fontsize	フォントサイズの変更
\adjustbaseline	ベースラインシフト量の設定
\{font|roman|kanji}encoding	エンコードの指定
\{font|roman|kanji}family	ファミリの指定
\{font|roman|kanji}series	シリーズの指定
\{font|roman|kanji}shape	シェイプの指定
\use{font|roman|kanji}	書体の切り替え
\normalfont	デフォルト値の設定に切り替える
\mcfamily,\gtfamily	和文書体を明朝体、ゴシック体にする
\textunderscore	テキストモードでの下線マクロ

\DeclareFontEncoding
\DeclareFontEncoding@

欧文エンコードを宣言するためのコマンドです。ltfssbas.dtx で定義されているものを、\fenc@list を作るように再定義をしています。

```
\def\DeclareFontEncoding{%
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareFontEncoding@}
%
```

```
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
     \def\cdp@elt{\noexpand\cdp@elt}%
     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                    {\default@family}{\default@series}%
                    {\default@shape}}%
     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
     \def\enc@elt{\noexpand\enc@elt}%
     \xdef\fenc@list{\fenc@list\enc@elt<#1>}%
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
  \xdef\LastDeclaredEncoding{#1}%
  }
```

\DeclareKanjiEncoding
\DeclareYokoKanjiEncoding
\DeclareYokoKanjiEncoding@
\DeclareTateKanjiEncoding
\DeclareTateKanjiEncoding@

和文エンコードの宣言をするコマンドです。

```
\def\DeclareKanjiEncoding#1{%
  \@latex@warning{%
     The \string\DeclareKanjiEncoding\space is obsoleted command.  Please use
     \MessageBreak
     the \string\DeclareTateKanjiEncoding\space for ‘Tate-kumi’ encoding, and
     \MessageBreak
     the \string\DeclareYokoKanjiEncoding\space for ‘Yoko-kumi’ encoding.
     \MessageBreak
     I treat the ‘#1’ encoding as ‘Yoko-kumi’.}
  \DeclareYokoKanjiEncoding{#1}%
}
\def\DeclareYokoKanjiEncoding{%
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareYokoKanjiEncoding@}
%
\def\DeclareYokoKanjiEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                   {\default@k@family}{\default@k@series}%
                   {\default@k@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@kcmd
    \def\enc@elt{\noexpand\enc@elt}%
    \xdef\kyenc@list{\kyenc@list\enc@elt<#1>}%
    \xdef\kenc@list{\kenc@list\enc@elt<#1>}%
  \else
    \@font@info{Redeclaring KANJI (yoko) font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@KM#3}%
  }
%
```

```
\def\DeclareTateKanjiEncoding{%
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareTateKanjiEncoding@}
%
\def\DeclareTateKanjiEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                   {\default@k@family}{\default@k@series}%
                   {\default@k@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@kcmd
    \def\enc@elt{\noexpand\enc@elt}%
    \xdef\ktenc@list{\ktenc@list\enc@elt<#1>}%
    \xdef\kenc@list{\kenc@list\enc@elt<#1>}%
  \else
    \@font@info{Redeclaring KANJI (tate) font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@KM#3}%
  }
%
\@onlypreamble\DeclareKanjiEncoding
\@onlypreamble\DeclareYokoKanjiEncoding
\@onlypreamble\DeclareYokoKanjiEncoding@
\@onlypreamble\DeclareTateKanjiEncoding
\@onlypreamble\DeclareTateKanjiEncoding@
```

\DeclareKanjiEncodingDefaults 和文エンコードのデフォルト値を宣言するコマンドです。

```
\def\DeclareKanjiEncodingDefaults#1#2{%
  \ifx\relax#1\else
    \ifx\default@KT\@empty\else
      \@font@info{Overwriting KANJI encoding scheme text defaults}%
    \fi
    \gdef\default@KT{#1}%
  \fi
  \ifx\relax#2\else
    \ifx\default@KM\@empty\else
      \@font@info{Overwriting KANJI encoding scheme math defaults}%
    \fi
    \gdef\default@KM{#2}%
  \fi}
\let\default@KT\@empty
\let\default@KM\@empty
\@onlypreamble\DeclareKanjiEncodingDefaults
```

\DeclareFontFamily 欧文ファミリを宣言するためのコマンドです。\ffam@list を作るように再定義をします。

```
\def\DeclareFontFamily#1#2#3{%
 \@ifundefined{T@#1}%
    {\@latex@error{Encoding scheme `#1' unknown}\@eha}%
```

```
    {\edef\tmp@item{{#2}}%
     \expandafter\expandafter\expandafter
     \inlist@\expandafter\tmp@item\expandafter{\ffam@list}%
     \ifin@ \else
        \def\fam@elt{\noexpand\fam@elt}%
        \xdef\ffam@list{\ffam@list\fam@elt<#2>}%
     \fi
     \def\reserved@a{#3}%
     \global
     \expandafter\let\csname #1+#2\expandafter\endcsname
            \ifx \reserved@a\@empty
              \@empty
            \else \reserved@a
            \fi
    }%
}
```

\DeclareKanjiFamily 欧文ファミリを宣言するためのコマンドです。

```
\def\DeclareKanjiFamily#1#2#3{%
 \@ifundefined{T@#1}%
    {\@latex@error{KANJI Encoding scheme `#1' unknown}\@eha}%
    {\edef\tmp@item{{#2}}%
     \expandafter\expandafter\expandafter
     \inlist@\expandafter\tmp@item\expandafter{\kfam@list}%
     \ifin@ \else
        \def\fam@elt{\noexpand\fam@elt}%
        \xdef\kfam@list{\kfam@list\fam@elt<#2>}%
     \fi
     \def\reserved@a{#3}%
     \global
     \expandafter\let\csname #1+#2\expandafter\endcsname
            \ifx \reserved@a\@empty
              \@empty
            \else \reserved@a
            \fi
     }%
}
```

\DeclareKanjiSubstitution
\DeclareErrorKanjiFont
目的の和文フォントが見つからなかったときに使うフォントの宣言をするコマンドです。それぞれ、\DeclareFontSubstitutionと\DeclareErrorFontに対応します。

```
\def\DeclareKanjiSubstitution#1#2#3#4{%
  \expandafter\ifx\csname T@#1\endcsname\relax
    \@latex@error{KANJI Encoding scheme `#1' unknown}\@eha
  \else
    \begingroup
       \def\reserved@a{#1}%
       \toks@{}%
       \def\cdp@elt##1##2##3##4{%
         \def\reserved@b{##1}%
         \ifx\reserved@a\reserved@b
           \addto@hook\toks@{\cdp@elt{#1}{#2}{#3}{#4}}%
         \else
           \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
```

```
        \fi}%
      \cdp@list
      \xdef\cdp@list{\the\toks@}%
    \endgroup
    \global\@namedef{D@#1}{\def\default@family{#2}%
                           \def\default@series{#3}%
                           \def\default@shape{#4}}%
  \fi}
%
\def\DeclareErrorKanjiFont#1#2#3#4#5{%
   \xdef\error@kfontshape{%
      \noexpand\expandafter\noexpand\split@name\noexpand\string
      \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
      \noexpand\@nil}%
   \gdef\default@k@family{#2}%
   \gdef\default@k@series{#3}%
   \gdef\default@k@shape{#4}%
   \global\let\k@family\default@k@family
   \global\let\k@series\default@k@series
   \global\let\k@shape\default@k@shape
   \gdef\f@size{#5}%
   \gdef\f@baselineskip{#5pt}}
%
\@onlypreamble\DeclareKanjiSubstitution
\@onlypreamble\DeclareErrorKanjiFont
```

\DeclareFixedFont　フォント名を宣言するコマンドです。

```
\def\DeclareFixedFont#1#2#3#4#5#6{%
   \begingroup
      \let\afont\font
      \math@fontsfalse
      \every@math@size{}%
      \fontsize{#6}\z@
      \edef\tmp@item{{#2}}%
      \expandafter\expandafter\expandafter
      \inlist@\expandafter\tmp@item\expandafter{\kyenc@list}%
      \ifin@
        \usekanji{#2}{#3}{#4}{#5}%
        \let\font\jfont
      \else
        \expandafter\expandafter\expandafter
        \inlist@\expandafter\tmp@item\expandafter{\ktenc@list}%
        \ifin@
          \usekanji{#2}{#3}{#4}{#5}%
          \let\font\tfont
        \else
          \useroman{#2}{#3}{#4}{#5}%
          \let\font\afont
        \fi
      \fi
      \global\expandafter\let\expandafter#1\the\font
      \let\font\afont
   \endgroup
  }
```

\reDeclareMathAlphabet 数式モード内で、数式文字用の和欧文フォントを同時に切り替えるコマンドです。

pLaTeX 2εには、本来の動作モードと 2.09 互換モードの二つがあり、両モードで数式文字を変更するコマンドや動作が異なります。本来の動作モードでは、\mathrm{...}のように\math??に引数を指定して使います。このときは引数にだけ影響します。2.09 互換モードでは、\rm のような二文字コマンドを使います。このコマンドには引数を取らず、書体はグルーピングの範囲で反映されます。二文字コマンドは、ネイティブモードでも使えるようになっていて、動作も 2.09 互換モードのコマンドと同じです。

しかし、内部的には\@math??という一つのコマンドがすべての動作を受け持ち、\math??コマンドや\??コマンドから呼び出された状態に応じて、動作を変えています。したがって、欧文フォントと和文フォントの両方を一度に変更する、数式文字変更コマンドを作るとき、それぞれの状態に合った動作で動くようにフォント切り替えコマンドを実行させる必要があります。

```
\def\reDeclareMathAlphabet#1#2#3{%
  \DeclareRobustCommand{#1}[1]{%
    \ifmmode
      \ifx\math@bgroup\@empty % 2.09 compatibility
        #2\relax#3\relax##1\relax
      \else % native mode
        \ifx\math@bgroup\relax % oldstyle
          #2\relax\@fontswitch\relax{#3}##1\relax
        \else
          #2{#3{##1}}%
        \fi
      \fi
    \else
      #2{##1}%
    \fi
  }%
}
\@onlypreamble\reDeclareMathAlphabet
```

\DeclareRelationFont \SetRelationFont 和文書体に対する従属書体を宣言するコマンドです。**従属書体**とは、ある和文書体とペアになる欧文書体のことです。主に多書体パッケージskfonts を用いるための仕組みです。

\DeclareRelationFont コマンドの最初の 4 つの引数の組が和文書体の属性、その後の 4 つの引数の組が従属書体の属性です。

```
\DeclareRelationFont{JY1}{mc}{m}{n}{OT1}{cmr}{m}{n}
\DeclareRelationFont{JY1}{gt}{m}{n}{OT1}{cmr}{bx}{n}
```

上記の例は、明朝体の従属書体としてコンピュータモダンローマン、ゴシック体の従属書体としてコンピュータモダンボールドを宣言しています。カレント和文書体が\JY1/mc/m/n となると、自動的に欧文書体が\OT1/cmr/m/n になります。また、和文書体が\JY1/gt/m/n になったときは、欧文書体が\OT1/cmr/bx/n になります。

和文書体のシェイプ指定を省略するとエンコード／ファミリ／シリーズの組合せで従属書体が使われます。このときは、\selectfont が呼び出された時点でのシェイプ（\f@shape）の値が使われます。

\DeclareRelationFont の設定値はグローバルに有効です。\SetRelationFont の設定値はローカルに有効です。フォント定義ファイルで宣言をする場合は、\DeclareRelationFont を使ってください。

```
\def\all@shape{all}%
\def\DeclareRelationFont#1#2#3#4#5#6#7#8{%
  \def\rel@shape{#4}%
  \ifx\rel@shape\@empty
     \global
     \expandafter\def\csname rel@#1/#2/#3/all\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}}%
  \else
     \global
     \expandafter\def\csname rel@#1/#2/#3/#4\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}\romanshape{#8}}%
  \fi
}
\def\SetRelationFont#1#2#3#4#5#6#7#8{%
  \def\rel@shape{#4}%
  \ifx\rel@shape\@empty
     \expandafter\def\csname rel@#1/#2/#3/all\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}}%
  \else
     \expandafter\def\csname rel@#1/#2/#3/#4\endcsname{%
       \romanencoding{#5}\romanfamily{#6}%
       \romanseries{#7}\romanshape{#8}}%
  \fi
}
```

\if@knjcmd
\userelfont

\if@knjcmd は欧文書体を従属書体にするかどうかのフラグです。このフラグが真になると、欧文書体に従属書体が使われます。このフラグは\userelfont コマンドによって、**真**となります。そして\selectfont 実行後には**偽**に初期化されます。

```
\newif\if@knjcmd
\def\userelfont{\@knjcmdtrue}
```

\selectfont

\selectfont のオリジナルからの変更部分は、次の 3 点です。

- 和文書体を変更する部分
- 従属書体に変更する部分
- 和欧文のベースラインを調整する部分

\selectfont コマンドは、まず、和文フォントを切り替えます。

```
⟨/plcore⟩
```

```
⟨*plcore | trace⟩
\DeclareRobustCommand\selectfont{%
  \let\tmp@error@fontshape\error@fontshape
  \let\error@fontshape\error@kfontshape
  \edef\tmp@item{{\k@encoding}}%
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kyenc@list}%
  \ifin@
    \let\cy@encoding\k@encoding
    \edef\ct@encoding{\csname t@enc@\k@encoding\endcsname}%
  \else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\ktenc@list}%
    \ifin@
      \let\ct@encoding\k@encoding
      \edef\cy@encoding{\csname y@enc@\k@encoding\endcsname}%
    \else
      \@latex@error{KANJI Encoding scheme `\k@encoding' unknown}\@eha
    \fi
  \fi
  \let\font\tfont
  \let\k@encoding\ct@encoding
  \xdef\font@name{\csname\curr@kfontshape/\f@size\endcsname}%
  \pickup@font
  \font@name
  \let\font\jfont
  \let\k@encoding\cy@encoding
  \xdef\font@name{\csname\curr@kfontshape/\f@size\endcsname}%
  \pickup@font
  \font@name
  \expandafter\def\expandafter\k@encoding\tmp@item
  \kenc@update
  \let\error@fontshape\tmp@error@fontshape
```

次に、\if@knjcmd が真の場合、欧文書体を現在の和文書体に関連付けされたフォントに変えます。このフラグは\userelfont コマンドによって真となります。このフラグはここで再び、偽に設定されます。

```
  \if@knjcmd \@knjcmdfalse
    \expandafter\ifx
    \csname rel@\k@encoding/\k@family/\k@series/\k@shape\endcsname\relax
      \expandafter\ifx
         \csname rel@\k@encoding/\k@family/\k@series/all\endcsname\relax
      \else
         \csname rel@\k@encoding/\k@family/\k@series/all\endcsname
      \fi
    \else
       \csname rel@\k@encoding/\k@family/\k@series/\k@shape\endcsname
    \fi
  \fi
```

そして、欧文フォントを切り替えます。

```
  \let\font\afont
  \xdef\font@name{\csname\curr@fontshape/\f@size\endcsname}%
  \pickup@font
```

```
  \font@name
⟨trace⟩\ifnum \tracingfonts>\tw@
⟨trace⟩  \@font@info{Roman:Switching to \font@name}\fi
  \enc@update
```

最後に、サイズが変更されていれば、ベースラインの調整などを行ないます。英語版の\selectfont では最初に行なっていますが、pLaTeX 2$_\varepsilon$ではベースラインシフトの調整をするために、書体を確定しなければならないため、一番最後に行ないます

```
  \ifx\f@linespread\baselinestretch \else
    \set@fontsize\baselinestretch\f@size\f@baselineskip
  \fi
  \size@update}
```

\KanjiEncodingPair 和文の縦横のエンコーディングはそれぞれ対にして扱うため、セット化します

```
\def\KanjiEncodingPair#1#2{\@namedef{t@enc@#1}{#2}\@namedef{y@enc@#2}{#1}}
\KanjiEncodingPair{JY1}{JT1}
```

\set@fontsize \fontsize コマンドの内部形式です。ベースラインの設定と、支柱の設定を行ないます。

```
\def\set@fontsize#1#2#3{%
    \@defaultunits\@tempdimb#2pt\relax\@nnil
    \edef\f@size{\strip@pt\@tempdimb}%
    \@defaultunits\@tempskipa#3pt\relax\@nnil
    \edef\f@baselineskip{\the\@tempskipa}%
    \edef\f@linespread{#1}%
    \let\baselinestretch\f@linespread
    \def\size@update{%
      \baselineskip\f@baselineskip\relax
      \baselineskip\f@linespread\baselineskip
      \normalbaselineskip\baselineskip
```

ここで、ベースラインシフトの調整と支柱を組み立てます。

```
      \adjustbaseline
      \setbox\strutbox\hbox{\yoko
          \vrule\@width\z@
                \@height.7\baselineskip \@depth.3\baselineskip}%
      \setbox\tstrutbox\hbox{\tate
          \vrule\@width\z@
                \@height.5\baselineskip \@depth.5\baselineskip}%
      \setbox\zstrutbox\hbox{\tate
          \vrule\@width\z@
                \@height.7\baselineskip \@depth.3\baselineskip}%
```

フォントサイズとベースラインに関する診断情報を出力します。

```
⟨*trace⟩
     \ifnum \tracingfonts>\tw@
       \ifx\f@linespread\@empty
         \let\reserved@a\@empty
       \else
         \def\reserved@a{\f@linespread x}%
       \fi
       \@font@info{Changing size to\space
```

```
392              \f@size/\reserved@a \f@baselineskip}%
393        \aftergroup\type@restoreinfo
394      \fi
395 ⟨/trace⟩
396       \let\size@update\relax}}
```

\adjustbaseline 現在の和文フォントの空白（EUC コード 0xA1A1）の中央に現在の欧文フォントの “/” の中央がくるようにベースラインシフトを設定します。

当初はまずベースラインシフト量をゼロにしていましたが、\tbaselineshift を連続して変更した後に鉤括弧類を使うと余計なアキがでる問題が起こるため、\tbaselineshift をゼロクリアする処理を削除しました。

しかし、それではベースラインシフトを調整済みの欧文ボックスと比較してしまうため、計算した値が大きくなってしまいます。そこで、このボックスの中でゼロにするようにしました。また、“/” と比較していたのを “M” にしました。

```
397 \newbox\adjust@box
398 \newdimen\adjust@dimen
399 \def\adjustbaseline{%
```

和文フォントの基準値を設定します。

```
400     \setbox\adjust@box\hbox{\char\euc"A1A1}%
401     \cht\ht\adjust@box
402     \cdp\dp\adjust@box
403     \cwd\wd\adjust@box
404     \cvs\normalbaselineskip
405     \chs\cwd
406     \cHT\cht \advance\cHT\cdp
```

基準となる欧文フォントの文字を含んだボックスを作成し、ベースラインシフト量の計算を行ないます。計算式は次のとおりです。

$$\text{ベースラインシフト量} \quad = \quad \{(\text{全角空白の深さ}) - (\text{/の深さ})\} \\ - \frac{(\text{全角空白の高さ} + \text{深さ}) - (\text{/の高さ} + \text{深さ})}{2}$$

```
407   \iftdir
408     \setbox\adjust@box\hbox{\tbaselineshift\z@ M}%
409     \adjust@dimen\ht\adjust@box
410     \advance\adjust@dimen\dp\adjust@box
411     \advance\adjust@dimen-\cHT
412     \divide\adjust@dimen\tw@
413     \advance\adjust@dimen\cdp
414     \advance\adjust@dimen-\dp\adjust@box
415     \tbaselineshift\adjust@dimen
416 ⟨trace⟩   \ifnum \tracingfonts>\tw@
417 ⟨trace⟩     \typeout{baselineshift:\the\tbaselineshift}
418 ⟨trace⟩   \fi
419   \fi}
420 ⟨/plcore | trace⟩
421 ⟨*plcore⟩
```

\romanencoding
\kanjiencoding
\fontencoding

書体のエンコードを指定するコマンドです。\fontencoding コマンドは和欧文のどちらかに影響します。\DeclareKanjiEncoding で指定されたエンコードは和文エンコードとして、\DeclareFontEncoding で指定されたエンコードは欧文エンコードとして認識されます。

\kanjiencoding と\romanencoding は与えられた引数が、エンコードとして登録されているかどうかだけを確認し、それが和文か欧文かのチェックは行なっていません。そのため、高速に動作をしますが、\kanjiencoding に欧文エンコードを指定したり、逆に\romanencoding に和文エンコードを指定した場合はエラーとなります。

```
\DeclareRobustCommand\romanencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{Encoding scheme `#1' unknown}\@eha
    \else
      \edef\f@encoding{#1}%
      \ifx\cf@encoding\f@encoding
        \let\enc@update\relax
      \else
        \let\enc@update\@@enc@update
      \fi
    \fi
}
\DeclareRobustCommand\kanjiencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{KANJI Encoding scheme `#1' unknown}\@eha
    \else
      \edef\k@encoding{#1}%
      \ifx\ck@encoding\k@encoding
         \let\kenc@update\relax
      \else
         \let\kenc@update\@@kenc@update
      \fi
    \fi
}
\DeclareRobustCommand\fontencoding[1]{%
  \edef\tmp@item{{#1}}%
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kenc@list}%
  \ifin@ \kanjiencoding{#1}\else\romanencoding{#1}\fi}
```

\@@kenc@update

\kanjiencoding コマンドのコードからもわかるように、\ck@encoding と\k@encoding が異なる場合、\kenc@update コマンドは\@@kenc@update コマンドと等しくなります。

\@@kenc@update コマンドは、そのエンコードでのデフォルト値を設定するためのコマンドです。欧文用の\@@enc@update コマンドでは、452 行目と 453 行目のような代入もしていますが、和文用にはコメントにしてあります。これらは\DeclareTextCommand や\ProvideTextCommand などでエンコードごとに設定されるコマンドを使うための仕組みです。しかし、和文エンコードに依存するようなコマンドやマクロを作成することは、現時点では、ないと思います。

```
\def\@@kenc@update{%
%  \expandafter\let\csname\ck@encoding -cmd\endcsname\@changed@kcmd
%  \expandafter\let\csname\k@encoding-cmd\endcsname\@current@cmd
  \default@KT
  \csname T@\k@encoding\endcsname
  \csname D@\k@encoding\endcsname
  \let\kenc@update\relax
  \let\ck@encoding\k@encoding
  \edef\tmp@item{{\k@encoding}}%
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kyenc@list}%
  \ifin@ \let\cy@encoding\k@encoding
  \else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\ktenc@list}%
    \ifin@ \let\ct@encoding\k@encoding
    \else
      \@latex@error{KANJI Encoding scheme `\k@encoding' unknown}\@eha
    \fi
  \fi
}
\let\kenc@update\relax
```

\@changed@cmd の和文エンコーディングバージョン。

```
\def\@changed@kcmd#1#2{%
   \ifx\protect\@typeset@protect
      \@inmathwarn#1%
      \expandafter\ifx\csname\ck@encoding\string#1\endcsname\relax
         \expandafter\ifx\csname ?\string#1\endcsname\relax
            \expandafter\def\csname ?\string#1\endcsname{%
               \TextSymbolUnavailable#1%
            }%
         \fi
         \global\expandafter\let
               \csname\cf@encoding \string#1\expandafter\endcsname
               \csname ?\string#1\endcsname
      \fi
      \csname\ck@encoding\string#1%
         \expandafter\endcsname
   \else
      \noexpand#1%
   \fi}
```

\@notkfam
\@notffam

\fontfamily コマンド内で使用するフラグです。@notkfam フラグは和文ファミリでなかったことを、@notffam フラグは欧文ファミリでなかったことを示します。

```
\newif\if@notkfam
\newif\if@notffam
```

```
\newif\if@tempswz
```

\romanfamily
\kanjifamily
\fontfamily

書体のファミリを指定するコマンドです。

\kanjifamily と \romanfamily は与えられた引数が、和文あるいは欧文のファミリとして正しいかのチェックは行なっていません。そのため、高速に動作をします

が、\kanjifamilyに欧文ファミリを指定したり、逆に\romanfamilyに和文ファミリを指定した場合は、エラーとなり、代用フォントかエラーフォントが使われます。

```
\DeclareRobustCommand\romanfamily[1]{\edef\f@family{#1}}
\DeclareRobustCommand\kanjifamily[1]{\edef\k@family{#1}}
```

\fontfamilyは、指定された値によって、和文ファミリか欧文ファミリ、**あるいは両方**のファミリを切り替えます。和欧文ともに無効なファミリ名が指定された場合は、和欧文ともに代替書体が使用されます。

引数が\rmfamilyのような名前で与えられる可能性があるため、まず、これを展開したものを作ります。

また、和文ファミリと欧文ファミリのそれぞれになかったことを示すフラグを偽にセットします。

```
\DeclareRobustCommand\fontfamily[1]{%
  \edef\tmp@item{{#1}}%
  \@notkfamfalse
  \@notffamfalse
```

次に、この引数が\kfam@listに登録されているかどうかを調べます。登録されていれば、\k@familyにその値を入れます。

```
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kfam@list}%
  \ifin@ \edef\k@family{#1}%
```

そうでないときは、\notkfam@listに登録されているかどうかを調べます。登録されていれば、この引数は和文ファミリではありませんので、\@notkfamフラグを真にして、欧文ファミリのルーチンに移ります。

このとき、\efam@listを調べるのではないことに注意をしてください。\efam@listを調べ、これにないファミリを和文ファミリであるとすると、たとえば、欧文ナールファミリが定義されているけれども、和文ナールファミリが未定義の場合、\fontfamily{nar}という指定は、narが\efam@listにだけ、登録されているため、和文書体をナールにすることができません。

逆に、\kfam@listに登録されていないからといって、\k@familyにnarを設定すると、cmrのようなファミリも\k@familyに設定される可能性があります。したがって、「欧文でない」を明示的に示す\notkfam@listを見る必要があります。

```
  \else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\notkfam@list}%
    \ifin@ \@notkfamtrue
```

\notkfam@listに登録されていない場合は、フォント定義ファイルが存在するかどうかを調べます。ファイルが存在する場合は、\k@familyを変更します。ファイルが存在しない場合は、\notkfam@listに登録します。

\kenc@listに登録されているエンコードと、指定された和文ファミリの組合せのフォント定義ファイルが存在する場合は、\k@familyに指定された値を入れます。

```
    \else
      \@tempswzfalse
```

```
      \def\fam@elt{\noexpand\fam@elt}%
      \message{(I search kanjifont definition file:}%
      \def\enc@elt<##1>{\message{.}%
        \edef\reserved@a{\lowercase{\noexpand\IfFileExists{##1#1.fd}}}%
        \reserved@a{\@tempswztrue}{}\relax}%
      \kenc@list
      \message{)}%
      \if@tempswz
        \edef\k@family{#1}%
```

つぎの部分が実行されるのは、和文ファミリとして認識できなかった場合です。この場合は、\@notkfam フラグを真にして、\notkfam@list に登録します。

```
      \else
        \@notkfamtrue
        \xdef\notkfam@list{\notkfam@list\fam@elt<#1>}%
      \fi
```

\kfam@list と\notkfam@list に登録されているかどうかを調べた\ifin@を閉じます。

```
  \fi\fi
```

欧文ファミリの場合も、和文ファミリと同様の方法で確認をします。

```
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\ffam@list}%
  \ifin@ \edef\f@family{#1}\else
    \expandafter\expandafter\expandafter
    \inlist@\expandafter\tmp@item\expandafter{\notffam@list}%
    \ifin@ \@notffamtrue \else
      \@tempswzfalse
      \def\fam@elt{\noexpand\fam@elt}%
      \message{(I search font definition file:}%
      \def\enc@elt<##1>{\message{.}%
        \edef\reserved@a{\lowercase{\noexpand\IfFileExists{##1#1.fd}}}%
        \reserved@a{\@tempswztrue}{}\relax}%
      \fenc@list
      \message{)}%
      \if@tempswz
        \edef\f@family{#1}%
      \else
        \@notffamtrue
        \xdef\notffam@list{\notffam@list\fam@elt<#1>}%
      \fi
  \fi\fi
```

最後に、指定された文字列が、和文ファミリと欧文ファミリのいずれか、あるいは両方として認識されたかどうかを確認します。

どちらとも認識されていない場合は、ファミリの指定ミスですので、代用フォントを使うために、故意に指定された文字列をファミリに入れます。

```
  \if@notkfam\if@notffam
      \edef\k@family{#1}\edef\f@family{#1}%
  \fi\fi}
```

\romanseries 書体のシリーズを指定するコマンドです。\fontseries コマンドは和欧文の両方に
\kanjiseries 影響します。
\fontseries
```
\DeclareRobustCommand\romanseries[1]{\edef\f@series{#1}}
\DeclareRobustCommand\kanjiseries[1]{\edef\k@series{#1}}
\DeclareRobustCommand\fontseries[1]{\kanjiseries{#1}\romanseries{#1}}
```

\romanshape 書体のシェイプを指定するコマンドです。\fontshape コマンドは和欧文の両方に
\kanjishape 影響します。
\fontshape
```
\DeclareRobustCommand\romanshape[1]{\edef\f@shape{#1}}
\DeclareRobustCommand\kanjishape[1]{\edef\k@shape{#1}}
\DeclareRobustCommand\fontshape[1]{\kanjishape{#1}\romanshape{#1}}
```

\usekanji 書体属性を一度に指定するコマンドです。和文書体には\usekanji を、欧文書体に
\useroman は\useroman を指定してください。
\usefont \usefont コマンドは、第一引数で指定されるエンコードによって、和文または欧文フォントを切り替えます。

```
\def\usekanji#1#2#3#4{%
    \kanjiencoding{#1}\kanjifamily{#2}\kanjiseries{#3}\kanjishape{#4}%
    \selectfont\ignorespaces}
\def\useroman#1#2#3#4{%
    \romanencoding{#1}\romanfamily{#2}\romanseries{#3}\romanshape{#4}%
    \selectfont\ignorespaces}
\def\usefont#1#2#3#4{%
  \edef\tmp@item{{#1}}%
  \expandafter\expandafter\expandafter
  \inlist@\expandafter\tmp@item\expandafter{\kenc@list}%
  \ifin@ \usekanji{#1}{#2}{#3}{#4}%
  \else\useroman{#1}{#2}{#3}{#4}%
  \fi}
```

\normalfont 書体をデフォルト値にするコマンドです。和文書体もデフォルト値になるように再定義しています。ただし高速化のため、\usekanji と\useroman を展開し、\selectfont を一度しか呼び出さないようにしています。

```
\DeclareRobustCommand\normalfont{%
    \kanjiencoding{\kanjiencodingdefault}%
    \kanjifamily{\kanjifamilydefault}%
    \kanjiseries{\kanjiseriesdefault}%
    \kanjishape{\kanjishapedefault}%
    \romanencoding{\encodingdefault}%
    \romanfamily{\familydefault}%
    \romanseries{\seriesdefault}%
    \romanshape{\shapedefault}%
    \selectfont\ignorespaces}
\adjustbaseline
\let\reset@font\normalfont
```

\mcfamily 和文書体を明朝体にする\mcfamily とゴシック体にする\gtfamily を定義します。
\gtfamily これらは、\rmfamily などに対応します。\mathmc と\mathgt は数式内で用いるときのコマンド名です。

```
\DeclareRobustCommand\mcfamily
        {\not@math@alphabet\mcfamily\mathmc
         \kanjifamily\mcdefault\selectfont}
\DeclareRobustCommand\gtfamily
        {\not@math@alphabet\gtfamily\mathgt
         \kanjifamily\gtdefault\selectfont}
```

\romanprocess@table
\kanjiprocess@table
\process@table

文書の先頭で、和文デフォルトフォントの変更が反映されないのを修正します。

```
\let\romanprocess@table\process@table
\def\kanjiprocess@table{%
  \kanjiencoding{\kanjiencodingdefault}%
  \kanjifamily{\kanjifamilydefault}%
  \kanjiseries{\kanjiseriesdefault}%
  \kanjishape{\kanjishapedefault}%
}
\def\process@table{%
  \romanprocess@table
  \kanjiprocess@table
}
\@onlypreamble\romanprocess@table
\@onlypreamble\kanjiprocess@table
```

\textunderscore

このコマンドはテキストモードで指定された_の内部コマンドです。縦組での位置を調整するように再定義をします。もとは ltoutenc.dtx で定義されています。

なお、_を数式モードで使うと\mathunderscore が実行されます。

```
\DeclareTextCommandDefault{\textunderscore}{%
  \leavevmode\kern.06em
  \iftdir\raise-\tbaselineshift\fi
  \vbox{\hrule\@width.3em}}
```

2.3 デフォルト設定ファイルの読み込み

最後に、デフォルト設定ファイルである、pldefs.ltx を読み込みます。このファイルについての詳細は、第 3 節を参照してください。TEX の入力ファイル検索パスに設定されているディレクトリに pldefs.cfg ファイルがある場合は、そのファイルを使います。

```
\InputIfFileExists{pldefs.cfg}
           {\typeout{*************************************^^J%
                     * Local config file pldefs.cfg used^^J%
                     *************************************}}%
           {\input{pldefs.ltx}}
```

⟨/plcore⟩

3 デフォルト設定ファイル

ここでは、フォーマットファイルに読み込まれるデフォルト値を設定しています。この節での内容は pldefs.ltx に出力されます。このファイルの内容を plcore.ltx に含めてもよいのですが、デフォルトの設定を参照しやすいように、別ファイルにしてあります。pldefs.ltx は plcore.ltx から読み込まれます。

プリロードサイズは、DOCSTRIP プログラムのオプションで変更することができます。これ以外の設定を変更したい場合は、pldefs.ltx を直接、修正するのではなく、このファイルを pldefs.cfg という名前でコピーをして、そのファイルに対して修正を加えるようにしてください。

```
607 ⟨*pldefs⟩
608 \ProvidesFile{pldefs.ltx}
609       [2000/07/13 v1.2 pLaTeX Kernel (Default settings)]
```

3.1 イタリック補正

\check@nocorr@ 「あ \texttt{abc}い」としたとき、書体の変更を指定された欧文の左側に和欧文間スペースが入らないのを修正します。

```
610 \def \check@nocorr@ #1#2\nocorr#3\@nil {%
611   \let \check@icl \relax% \maybe@ic から変更
612   \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
613   \def \reserved@a {\nocorr}%
614   \def \reserved@b {#1}%
615   \def \reserved@c {#3}%
616   \ifx \reserved@a \reserved@b
617     \ifx \reserved@c \@empty
618       \let \check@icl \@empty
619     \else
620       \let \check@icl \@empty
621       \let \check@icr \@empty
622     \fi
623   \else
624     \ifx \reserved@c \@empty
625     \else
626       \let \check@icr \@empty
627     \fi
628   \fi
629 }
```

3.2 テキストフォント

テキストフォントのための属性やエラー書体などの宣言です。

縦横エンコード共通：

```
630 \DeclareKanjiEncodingDefaults{}{}
631 \DeclareErrorKanjiFont{JY1}{mc}{m}{n}{10}
```

横組エンコード：

```
632 \DeclareYokoKanjiEncoding{JY1}{}{}
633 \DeclareKanjiSubstitution{JY1}{mc}{m}{n}
```

縦組エンコード：

```
634 \DeclareTateKanjiEncoding{JT1}{}{}
635 \DeclareKanjiSubstitution{JT1}{mc}{m}{n}
```

フォント属性のデフォルト値：

```
636 \newcommand\mcdefault{mc}
637 \newcommand\gtdefault{gt}
```

```
\newcommand\kanjiencodingdefault{JY1}
\newcommand\kanjifamilydefault{\mcdefault}
\newcommand\kanjiseriesdefault{\mddefault}
\newcommand\kanjishapedefault{\updefault}
```

和文エンコードの指定：

```
\kanjiencoding{JY1}
```

フォント定義：これらの具体的な内容は第 4 節を参照してください。

```
\input{jy1mc.fd}
\input{jy1gt.fd}
\input{jt1mc.fd}
\input{jt1gt.fd}
```

フォントを有効にする

```
\fontencoding{JT1}\selectfont
\fontencoding{JY1}\selectfont
```

\textmc
\textgt

テキストファミリを切り替えるためのコマンドです。ltfntcmd.dtx で定義されている\textrm などに対応します。

```
\DeclareTextFontCommand{\textmc}{\mcfamily}
\DeclareTextFontCommand{\textgt}{\gtfamily}
```

\em
\emph

従来は\em, \emph で和文フォントの切り替えは行っていませんでしたが、和文フォントも\gtfamily に切り替えるようにしました.

```
\DeclareRobustCommand\em
        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
                        \mcfamily \upshape \else \gtfamily \itshape \fi}
```

3.3 プリロードフォント

あらかじめフォーマットファイルにロードされるフォントの宣言です。DOCSTRIP プログラムのオプションでロードされるフォントのサイズを変更することができます。platex.ins ではxpt を指定しています。

```
⟨*xpt⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}{5,7,10,12}
\DeclarePreloadSizes{JY1}{gt}{m}{n}{5,7,10,12}
\DeclarePreloadSizes{JT1}{mc}{m}{n}{5,7,10,12}
\DeclarePreloadSizes{JT1}{gt}{m}{n}{5,7,10,12}
⟨/xpt⟩
⟨*xipt⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}{5,7,10.95,12}
\DeclarePreloadSizes{JY1}{gt}{m}{n}{5,7,10.95,12}
\DeclarePreloadSizes{JT1}{mc}{m}{n}{5,7,10.95,12}
\DeclarePreloadSizes{JT1}{gt}{m}{n}{5,7,10.95,12}
⟨/xipt⟩
⟨*xiipt⟩
\DeclarePreloadSizes{JY1}{mc}{m}{n}{7,9,12,14.4}
\DeclarePreloadSizes{JY1}{gt}{m}{n}{7,9,12,14.4}
\DeclarePreloadSizes{JT1}{mc}{m}{n}{7,9,12,14.4}
\DeclarePreloadSizes{JT1}{gt}{m}{n}{7,9,12,14.4}
```

```
671 ⟨/xiipt⟩
672 ⟨*ori⟩
673 \DeclarePreloadSizes{JY1}{mc}{m}{n}
674         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
675 \DeclarePreloadSizes{JY1}{gt}{m}{n}
676         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
677 \DeclarePreloadSizes{JT1}{mc}{m}{n}
678         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
679 \DeclarePreloadSizes{JT1}{gt}{m}{n}
680         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
681 ⟨/ori⟩
```

3.4 組版パラメータ

禁則パラメータや文字間へ挿入するスペースの設定などです。実際の各文字への禁則パラメータおよびスペースの挿入の許可設定などは、kinsoku.tex で行なっています。具体的な設定については、kinsoku.dtx を参照してください。

```
682 \InputIfFileExists{kinsoku.tex}%
683   {\message{Loading kinsoku patterns for japanese.}}
684   {\errhelp{The configuration for kinsoku is incorrectly installed.^^J%
685             If you don't understand this error message you need
686             to seek^^Jexpert advice.}%
687    \errmessage{OOPS! I can't find any kinsoku patterns for japanese^^J%
688                \space Think of getting some or the
689                platex2e setup will never succeed}\@@end}
```

組版パラメータの設定をします。\kanjiskip は、漢字と漢字の間に挿入されるグルーです。\noautospacing で、挿入を中止することができます。デフォルトは \autospacing です。

```
690 \kanjiskip=0pt plus .4pt minus .5pt
691 \autospacing
```

\xkanjiskip は、和欧文間に自動的に挿入されるグルーです。\noautoxspacing で、挿入を中止することができます。デフォルトは\autoxspacing です。

```
692 \xkanjiskip=.25zw plus1pt minus1pt
693 \autoxspacing
```

\jcharwidowpenalty は、パラグラフに対する禁則です。パラグラフの最後の行が 1 文字だけにならないように調整するために使われます。

```
694 \jcharwidowpenalty=500
```

最後に、\inhibitglue の簡略形を定義します。このコマンドは、和文フォントのメトリック情報から、自動的に挿入されるグルーの挿入を禁止します。

```
695 \def\<{\inhibitglue}
```

ここまでが、pldefs.ltx の内容です。

```
696 ⟨/pldefs⟩
```

4　フォント定義ファイル

ここでは、フォント定義ファイルの設定をしています。フォント定義ファイルは、LaTeX のフォント属性を TeX フォントに置き換えるためのファイルです。記述方法についての詳細は、fntguide.tex を参照してください。

欧文書体の設定については、cmfonts.fdd や slides.fdd などを参照してください。skfonts.fdd には、写研代用書体を使うためのパッケージとフォント定義が記述されています。

```
⟨JY1mc⟩\ProvidesFile{jy1mc.fd}
⟨JY1gt⟩\ProvidesFile{jy1gt.fd}
⟨JT1mc⟩\ProvidesFile{jt1mc.fd}
⟨JT1gt⟩\ProvidesFile{jt1gt.fd}
⟨JY1mc, JY1gt, JT1mc, JT1gt⟩        [1997/01/24 v1.3 KANJI font defines]
```

横組用、縦組用ともに、明朝体のシリーズbx がゴシック体となるように宣言しています。

```
⟨*JY1mc⟩
\DeclareKanjiFamily{JY1}{mc}{}
\DeclareRelationFont{JY1}{mc}{m}{}{OT1}{cmr}{m}{}
\DeclareRelationFont{JY1}{mc}{bx}{}{OT1}{cmr}{bx}{}
\DeclareFontShape{JY1}{mc}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*min
    <10.95><12><14.4><17.28><20.74><24.88> min10
    <-> min10
    }{}
\DeclareFontShape{JY1}{mc}{bx}{n}{<->ssub*gt/m/n}{}
⟨/JY1mc⟩
⟨*JT1mc⟩
\DeclareKanjiFamily{JT1}{mc}{}
\DeclareRelationFont{JT1}{mc}{m}{}{OT1}{cmr}{m}{}
\DeclareRelationFont{JT1}{mc}{bx}{}{OT1}{cmr}{bx}{}
\DeclareFontShape{JT1}{mc}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*tmin
    <10.95><12><14.4><17.28><20.74><24.88> tmin10
    <-> tmin10
    }{}
\DeclareFontShape{JT1}{mc}{bx}{n}{<->ssub*gt/m/n}{}
⟨/JT1mc⟩
⟨*JY1gt⟩
\DeclareKanjiFamily{JY1}{gt}{}
\DeclareRelationFont{JY1}{gt}{m}{}{OT1}{cmr}{bx}{}
\DeclareFontShape{JY1}{gt}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*goth
    <10.95><12><14.4><17.28><20.74><24.88> goth10
    <-> goth10
    }{}
\DeclareFontShape{JY1}{gt}{bx}{n}{<->ssub*gt/m/n}{}
⟨/JY1gt⟩
⟨*JT1gt⟩
\DeclareKanjiFamily{JT1}{gt}{}
\DeclareRelationFont{JT1}{gt}{m}{}{OT1}{cmr}{bx}{}
\DeclareFontShape{JT1}{gt}{m}{n}{<5> <6> <7> <8> <9> <10> sgen*tgoth
    <10.95><12><14.4><17.28><20.74><24.88> tgoth10
    <-> tgoth10
```

```
737     }{}
738 \DeclareFontShape{JT1}{gt}{bx}{n}{<->ssub*gt/m/n}{}
739 ⟨/JT1gt⟩
```