Pakiet Polski
wersja 1.3.4

Mariusz Olko
Litteræ
Górczewska 94/96/7
01–117 Warszawa
M.Olko@Litterae.com.pl

Marcin Woliński
wolinski@gust.org.pl

4 maja 2017

Spis treści

1 Informacje dla użytkowników

1.1 Krótki obraz całości

Paczka dystrybucyjna pakietu POLSKI składa się z kilku zasadniczych części.

- Najważniejszą częścią jest sam pakiet `polski.sty`, który dostarcza wszystkich (?) elementów potrzebnych do składu w języku polskim. Pozwala na stosowanie w różnych środowiskach, z polskimi wzorcami przenoszenia i bez, z polskimi czcionkami i bez nich. Posiada też możliwość upodobnienia się na poziomie poleceń w 99% do LaMeXa. Szczegółowy opis pakietu znajduje się w następnym rozdziale.

- Drugą składową stanowią pliki opisu czcionek mające standardowo rozszerzenie `.fd`, a generowane przez program DocStrip z pliku `plfonts.fdd`. Znajdują się w nich informacje na temat fontów `PL` czyli polskich wersji fontów Computer Modern (dystrybuowanych z LaMeXem) oraz czcionek `PC` czyli polskich wersji czcionek Computer Concrete. Dzięki zawartym tam informacjom czcionki te stają się dostępne w Nowym Mechaniźmie Wyboru Fontów (*ang.* New Font Selection Scheme). Z pliku `plfonts.fdd` można wygenerować pliki opisu fontów zarówno w Starym Układzie (OT1) jak i w Układzie Polskim (OT4).

- Ostatni element to dwa dodatkowe pakiety, wspomagające pracę w nietypowych warunkach: PLPREFIX i OT1PATCH. Ich opis można znaleźć w nich samych.

Kod pakietu POLSKI bazuje na rozwiązaniach zastosowanych w formatach MeX/LaMeX autorstwa Marka Ryćko i Bogusława Jackowskiego.

1.2 Jak działa pakiet POLSKI

Po załadowaniu pakietu zmienione zostają wewnętrzne kody TeXa dla odnośnych liter polskiego alfabetu w Nowym Układzie (T1). Te zmiany pozwalają na definiowanie makrokomend, które mają w nazwie polskie litery, umożliwiają prawidłową zamianę liter małych na duże, a także pozwalają algorytmowi przenoszenia wyrazów traktować polskie litery jako litery. Następnie podjęte zostaje poszukiwanie polskich wzorców przenoszenia i ich uaktywnienie. Jeżeli wzorce przenoszenia nie zostaną znalezione, pakiet POLSKI wypisuje komunikat o błędzie i blokuje przenoszenie wyrazów.

W kolejnym kroku zdefiniowana zostaje notacja „ciachowa". Notacja ta, wprowadzona w MeXu przez Ryćkę i Jackowskiego, pozwala na zapisywanie polskich

liter w postaci dwóch znaków *ukośnik* oraz *litera*. Taki zapis pozwala na przesyłanie tekstów pocztą elektroniczną oraz na pracę w miejscach gdzie nie ma wbudowanego w system wsparcia dla języka polskiego (niektóre bardzo stare instalacje `UNIX`owe). Pakiet POLSKI uzyskuje wszystkie polskie litery zdefiniowane w standardowym TEXu (tj. *ó, ż* czy *ł*) za pomocą standardowych makr TEXa (tzn. np. `\'o`, `\.z` czy też `\l`), natomiast litery takie jak *ą* czy *ę* za pomocą standardowego makra LATEXowego `\k`. Cała dalsza łączność pomiędzy komendą *ciach litera* a wydrukowanym znakiem jest zapewniona poprzez definicje układów czcionek. To właśnie w tych plikach jest zdefiniowane, że np. w Starym Układzie (OT1) literę *ó* otrzymuje się przez złożenie akcentu ´ oraz litery *o* natomiast w Układach Nowym (T1) oraz Polskim (OT4) przez postawienie znaku o kodzie 161. Daje to dużą elastyczność i pozwala na bardzo łatwe użycie czcionek w dowolnym sensownym układzie. Do korzystania z czcionek `pl` zdefiniowany został nowy układ czcionek nazwany `OT4`. Szczegółowe informacje o funkcjonowania układów czcionek można znaleźć w plikach standardowej dystrybucji LATEXa `ltoutenc.dtx` oraz `fntguide.tex`.

Pakiet POLSKI pozwala na skład z różnymi zestawami czcionek w różnych układach. Początkowy układ czcionek dokumentu może zostać wybrany przez dodanie do wywołania pakietu odpowiednich opcji (patrz 1.3) lub użycie standardowego pakietu `fontenc`. Jeśli jednak nie zmieniono początkowego układu, pakiet POLSKI próbuje odszukać w systemie plik `ot4cmr.fd`, zawierający LATEXowe opisy czcionek `pl`. Jeżeli taki plik zostanie znaleziony, czynione jest założenie, że w systemie zainstalowane są również same czcionki `pl` i pakiet zmienia początkowy układ na `OT4`. Jeśli plik nie zostanie odszukany, to układ pozostaje bez zmian.

Pakiet POLSKI przedefiniowuje wszystkie napisy, które mogą pojawić się wygenerowane automatycznie przez LATEXa, takie jak: rozdział, spis treści itp. Zmieniona zostaje też definicja makra `\today` tak, aby data była drukowana po polsku. Ponieważ w niektórych sytuacjach na końcu daty pisze się całe słowo „roku", czasami tylko samą literę „r.", a czasami nic, wprowadzone zostało makro `\PLdateending`, które rozwija się zaraz za rokiem i w razie potrzeby może zostać łatwo przedefiniowane. Co więcej zachowanie makra `\today` można zmienić za pomocą następujących opcji pakietu: `roku`, `r.`, `noroku` (ta ostatnia jest domyślna, więc domyślnie po numerze roku nie jest nic dodawane).

Dodatkowo pakiet definiuje makro `\dywiz`, które pozwala na poprawne przeniesienie wyrazów złożonych zapisanych jako `biało\dywiz czerwony` i dzielonych jako

biało-
-czerwony.

Kolejny problem to pauzy (myślniki). Według polskich zwyczajów myślnik powinien być otoczony odstępami wielkości 2pt, oraz nie należy rozpoczynać wiersza tekstowego myślnikiem. Zalecenia te realizuje makro `\pauza`. Makro to zawiera w sobie potrzebne odstępy, należy więc go używać następująco:

```
Było zbyt ciemno\pauza powiedziała.
```

Uwaga: definicję tego makra traktujemy jako prowizoryczną. Może ulec zmianie!

W polskich zwyczajach typograficznych odstęp po kropce między zdaniami powinien być taki sam jak pomiędzy wyrazami w środku zdania, dlatego pakiet woła makro `\frenchspacing`.

Pewne zmiany dotyczą również matematyki. Najważniejszymi różnicami w składzie pomiędzy matematycznymi wydawnictwami polskimi i angielskojęzycznymi jest inny kształt znaków *mniejsze-równe* i *większe-równe* oraz inne skróty stosowane na oznaczenie tangensa, cotangensa i funkcji transcendentalnych. Zmianą kształtu znaków *mniejsze-równe* i *większe-równe* jest dokonywana wtedy, jeśli dostępne są matematyczne czcionki `pl`. Standardowo pakiet POLSKI definiuje nowe makra `\tg`, `\tgh`, `\ctg`, `\ctgh`, `\arc` oraz `\nwd` a następnie — uwaga — zmienione zostają symbole drukowane przez standardowe makra LaTeXa `\tan`, `\cot`, `\tanh`, `\coth`, `\arcsin`, `\arccos`, `\arctan`, `\gcd`. Dla pełności jest też definiowane makro `\arccot`, którego z tajemniczych przyczyn nie ma w wersji oryginalnej. Przedefiniowanie tych makr pozwala na cytowanie tych samych wzorów w pracy polskiej i angielskiej bez konieczności zmieniania ich zapisu. Standardowe symbole są zmieniane ponieważ LaTeX (czy też TeX) dawno przestał być tylko systemem składu. Stał się obecnie językiem, w którym zapisywane są wzory matematyczne i jest bardzo ważne jest aby, jeśli jest to możliwe, nie zmieniać „standardu" zapisu tego języka, lecz co najwyżej dostosowywać sposób w jaki jest on prezentowany na wydruku.

1.3 Dołączenie pakietu POLSKI do dokumentu

Pakiet POLSKI jest ładowany przez umieszczenie w preambule dokumentu zlecenia

```
\usepackage[opcje]{polski}
```

Użycie w wywołaniu pakietu opcji pozwala na dopasowanie jego zachowania do istniejącego środowiska i potrzeb.

OT1 świadomie nie chcemy zmieniać układu czcionek z układu podstawowego wbudowanego w LaTeXa.

OT4 przełącza układ czcionek na polski (OT4). Oznacza to, że w dokumencie będą wykorzystane czcionki `pl`.

T1 zmienia układ czcionek na Nowy Układ Czcionek. Opcja jest wygodna np. w połączeniu z pakietem czcionek PostScriptowych w układzie T1.

QX zmienia układ czcionek na QX. Użyteczna przy składzie fontami produkcji JNS Team: TeX Gyre Termes, TeX Gyre Heros, itd.

plmath przełącza czcionki matematyczne na `pl`, tzn. przedefiniowuje alfabety matematyczne i zestawy symboli. Dodatkowo zmienia LaTeXową definicję symboli większe-równe oraz mniejsze-równe.

nomathsymbols blokuje spolonizowanie przez pakiet znaczenia standardowych LaTeXowych symboli określających funkcje trygonometryczne oraz relacje większe-równe i mniejsze-równe

prefixinginverb powoduje, że notacja prefiksowa nie jest wyłączana w obrębie środowiska `verbatim` i w argumencie polecenia `\verb`. (Domyślnie aktywna).

noprefixinginverb powoduje, że notacja prefiksowa jest wyłączana w tych kontekstach.

MeX jest to tryb 100% zgodności z MEXem. Ta opcja definiuje wszystkie makra, które są normalnie dostępne dla użytkownika w MEXu. Pozwala to na kompilację dokumentów MEXowych bez dokonywania żadnych zmian.

Jeżeli nie użyto żadnej z opcji wyboru układu fontów, `polski.sty` próbuje włączyć fonty PL, jeżeli są one zainstalowane. Dotyczy to zarówno fontów tekstowych, jak i matematycznych. W instalacji zawierającej fonty PL wywołanie pakietu bez opcji jest równoważne wywołaniu

```
\usepackage[OT4,plmath]{polski}
```

Opcja `OT1` służy do powiedzenia pakietowi, że użytkownik świadomie używa układu nie zawierającego kompletu znaków potrzebnych do składu po polsku.

Dalsza część dokumentu opisuje kod samego pakietu oraz plików potrzebnych do instalacji czcionek polskich i wzorców przenoszenia w LATEXu. Dokumentacja jest w języku angielskim.

2 Source code of `polski.sty`

2.1 Writing banners

This package should work only with LATEX 2_ε, so we make sure the appropriate message is displayed when another TEX format is used.

```
⟨*style⟩
\NeedsTeXFormat{LaTeX2e}[1996/12/01]
```

Announce the name of the package to the world

```
\ProvidesPackage{polski}[2017/05/04 v1.3.4 Polish language package]
```

2.2 Category codes and all that

The settings described in this section are not appropriate for Unicode aware TEX engines. Hence a check whether the engine interprets the letter aogonek (2 bytes in UTF8) as a single entity:

```
\begingroup
\def\t#1#2!{\def\s{#2}}\t !%
\expandafter\endgroup\ifx\s\empty\else
```

Here we will define the codes for Polish diacritical characters. There are several codes we need to set for each of them. The most important one is the category code (`catcode`), which identifies the character as a letter to TEX. Other codes to set are lowercase and uppercase equivalents (`lccode` and `uccode`) used to determine the

proper character when lower and upper casing the string. These are now properly set in the kernel.

```
\@ifpackageloaded{inputenc}{\typeout{\space\space\space
                      Inputenc package detected. Catcodes not changed.}}{%
\catcode`\^^a1=11 %\lccode`\^^a1=`\^^a1 \uccode`\^^a1=`\^^81  % a ogonek
\catcode`\^^a2=11 %\lccode`\^^a2=`\^^a2 \uccode`\^^a2=`\^^82  % c acute
\catcode`\^^a6=11 %\lccode`\^^a6=`\^^a6 \uccode`\^^a6=`\^^86  % e ogonek
\catcode`\^^aa=11 %\lccode`\^^aa=`\^^aa \uccode`\^^aa=`\^^8a  % l crossed
\catcode`\^^ab=11 %\lccode`\^^ab=`\^^ab \uccode`\^^ab=`\^^8b  % n acute
\catcode`\^^f3=11 %\lccode`\^^f3=`\^^f3 \uccode`\^^f3=`\^^d3  % o acute
\catcode`\^^b1=11 %\lccode`\^^b1=`\^^b1 \uccode`\^^b1=`\^^91  % s acute
\catcode`\^^bb=11 %\lccode`\^^bb=`\^^bb \uccode`\^^bb=`\^^9b  % z dot
\catcode`\^^b9=11 %\lccode`\^^b9=`\^^b9 \uccode`\^^b9=`\^^99  % z acute
```

Now the same for uppercase letters.

```
\catcode`\^^81=11 %\lccode`\^^81=`\^^a1 \uccode`\^^81=`\^^81  % A ogonek
\catcode`\^^82=11 %\lccode`\^^82=`\^^a2 \uccode`\^^82=`\^^82  % C accute
\catcode`\^^86=11 %\lccode`\^^86=`\^^a6 \uccode`\^^86=`\^^86  % E ogonek
\catcode`\^^8a=11 %\lccode`\^^8a=`\^^aa \uccode`\^^8a=`\^^8a  % L crossed
\catcode`\^^8b=11 %\lccode`\^^8b=`\^^ab \uccode`\^^8b=`\^^8b  % N accute
\catcode`\^^d3=11 %\lccode`\^^d3=`\^^f3 \uccode`\^^d3=`\^^d3  % O acute
\catcode`\^^91=11 %\lccode`\^^91=`\^^b1 \uccode`\^^91=`\^^91  % S acute
\catcode`\^^9b=11 %\lccode`\^^9b=`\^^bb \uccode`\^^9b=`\^^9b  % Z dot
\catcode`\^^99=11 %\lccode`\^^99=`\^^b9 \uccode`\^^99=`\^^99  % Z acute
}
```

We finish by setting space factor codes (`sfcode`) for uppercase letters. When French spacing is turned off, TEX treats interword spacing after full stop in a special manner. If the last character before the period is lowercase letter then TEX assumes it is the end of the sentence, and makes the space wider (and more stretchable). However, if the last letter is uppercase, then TEX assumes it is an abbreviation and doesn't widen the space. (This is not the whole truth. Consult the TEXbook pages 285–287 for details.) We set `sfcode` for Polish capital letters.

```
\sfcode`\^^81=999    % A ogonek
\sfcode`\^^82=999    % C acute
\sfcode`\^^86=999    % E ogonek
\sfcode`\^^8a=999    % L crossed
\sfcode`\^^8b=999    % N acute
\sfcode`\^^d3=999    % O acute
\sfcode`\^^91=999    % S acute
\sfcode`\^^9b=999    % Z dot
\sfcode`\^^99=999    % Z acute
```

This provides for `\mathit` and friends to work correctly for Polish characters (when used with TCX).

```
\DeclareMathSymbol{^^a1}{\mathalpha}{letters}{`^^a1}
\DeclareMathSymbol{^^a2}{\mathalpha}{letters}{`^^a2}
\DeclareMathSymbol{^^a6}{\mathalpha}{letters}{`^^a6}
\DeclareMathSymbol{^^aa}{\mathalpha}{letters}{`^^aa}
\DeclareMathSymbol{^^ab}{\mathalpha}{letters}{`^^ab}
```

```
\DeclareMathSymbol{^^f3}{\mathalpha}{letters}{`^^f3}
\DeclareMathSymbol{^^b1}{\mathalpha}{letters}{`^^b1}
\DeclareMathSymbol{^^bb}{\mathalpha}{letters}{`^^bb}
\DeclareMathSymbol{^^b9}{\mathalpha}{letters}{`^^b9}
\DeclareMathSymbol{^^81}{\mathalpha}{letters}{`^^81}
\DeclareMathSymbol{^^82}{\mathalpha}{letters}{`^^82}
\DeclareMathSymbol{^^86}{\mathalpha}{letters}{`^^86}
\DeclareMathSymbol{^^8a}{\mathalpha}{letters}{`^^8a}
\DeclareMathSymbol{^^8b}{\mathalpha}{letters}{`^^8b}
\DeclareMathSymbol{^^d3}{\mathalpha}{letters}{`^^d3}
\DeclareMathSymbol{^^91}{\mathalpha}{letters}{`^^91}
\DeclareMathSymbol{^^9b}{\mathalpha}{letters}{`^^9b}
\DeclareMathSymbol{^^99}{\mathalpha}{letters}{`^^99}
\fi
```

2.3 Hyphenation

`\selecthyphenation` Here we define the hyphenation selecting operator. If a set of hyphenation patterns for a particular language is unavaiable, hyphenation in that language is turned off. For that we use following trick: a new language is allocated with no hyphenation patterns. Then switching to this language effectively switches hyphenation off (many thanks to Marek Ryćko).

```
\ifx\l@nohyphenation\@undefined
        \newlanguage\l@nohyphenation
\fi
\def\selecthyphenation#1{%
  \expandafter\ifx\csname l@#1\endcsname\relax
    \PackageError{polski}{No hyphenation patterns for language `#1'}
      {Hyphenation in this language will be disabled.}%
    \selecthyphenation{nohyphenation}%
  \else
    \language\csname l@#1\endcsname
  \fi
  }
```

At some point in time luaLaTeX stopped preloading hyphenation patterns in the format file. In case of luaLaTeX we try to load the hyphenation patterns for Polish at runtime. The first `\ifx` check for `\directlua`, which is characteristic for lua engine. An older version of luaLaTeX could have Polish patterns preloaded, so we check for `\l@polish` being defined.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
\else
\expandafter\ifx\csname l@polish\endcsname\relax
\newlanguage\l@polish
\language\l@polish
\InputIfFileExists{hyph-pl}{}{%
  \PackageError{polski}{Couldn't load hyphenation patterns for Polish}%
```

```
  {Missing file hyph-pl.tex from the hyph-utf8 project.}%
}%
\fi\fi
```

We try to switch to polish hyphenation patterns looking either for patterns name used by `hyphen.cfg` from old versions of POLSKI bundle or for new Babel-like name.

```
\ifx\polish\undefined
  \selecthyphenation{polish}
\else
    \language\polish
\fi
\lefthyphenmin=2
\righthyphenmin=2
```

2.4 Slash notation

The slash notation was introduced in the macro package LaTeX by Bogusław Jackowski and Marek Ryćko. It has been used since then in many places and became Polish TeX User's Group GUST "standard". What follows is the implementation of active slash or Polish slash macro.

`\Slash` We start by storing slash character (catcode 12 meaning `<other>`) in apropriately named macro.

```
\def\Slash{/}
```

`\PLSlash` Now we define macro `\PLSlash` which will actually be used in input files to access polish letters. It does not need to be robust. If it is, it breaks kerns (pointed out by Marcin Woliński).

```
\def\PLSlash#1{%
```

The first thing we do is to check whether the slash character is followed by an allowed character. The first test is for the second slash (or macro `\PLSlash`), in which case we just return *slash* character with category code `<other>`.

```
  \ifx#1\PLSlash
    \ifx\protect\@typeset@protect\else\protect\string\fi\Slash
  \else
```

If it was not a slash we test for a letter. We assume that there are defined macros which expand to the current definitions of Polish letters. We will give them names `\PLSlash@<character>`, so now we look if it is defined. If comparison with `\relax` is true the macro is not defined. We issue an error message with some help.

```
    \expandafter \ifx \csname PLSlash@\string#1\endcsname \relax
      \PLSlash@error#1%
    \else
```

If we got here, we can now expand polish character. However, we do that after completing all `\if`s.

```
      \expandafter\expandafter\expandafter\PLSlash@letter
      \expandafter\expandafter\expandafter#1%
    \fi
```

```
  \fi
}

\def\PLSlash@error#1{\PackageError{polski}{%
Illegal pair of characters /\noexpand#1 occurred}{%
Only a character from the set [acelnosxzACELNOSXZ,'<>/-]
                                        can appear after \Slash.\MessageBreak
Proceed, I will omit both \Slash\ and the character following it.\MessageBreak
You can also correct your mistake NOW, typing I followed by\MessageBreak
whatever should be in the place of the offending pair.}}
```

`\PlPrIeC` This macro is needed to protect against removing white space in TOC by Polish characters that have definition ending with a macro call (`\l` and `\L`). The macro is identical to `\IeC` from inputenc package, but we have to define it here not to depend on inputenc. The name is different not to cause conflict in case inputenc is loaded after plprefix.

```
\def\PlPrIeC{%
  \ifx\protect\@typeset@protect
    \expandafter\@firstofone
  \else
    \noexpand\PlPrIeC
  \fi
}
```

`\PLSlash@letter` This macro is very simple: it just invokes another macro with some wild name.

```
\def\PLSlash@letter#1{\csname PLSlash@#1\endcsname}
```

Next come the definitions of all Polish diacritics and special symbols. For each "slashed" character we define a macro expanding to its proper definition. Polish characters are defined as normal accented letters, and we expect that they will expand according to their definitions in the current font encoding. This allows us to use the same slash notation with any (decent) font encoding. For example T1 and OT4 encodings will use letters, but OT1 will do what it can —ie. insert simple accented characters (with `a` and `e` left untouched). For more information on the work of encoding engine consult LaTeX file `ltoutenc.dtx`.

The following macro is just a helper which will be undefined after use.

```
\def\PL@accent@def#1#2{%
  \expandafter\def \csname PLSlash@\string #1\endcsname{#2}}
```

The real definition will take place at the beginning of the document. This is small optimization. We assume that the encoding at this stage is what will be default for the rest of the document. If document starts in OT1 encoding we warn user that he can loose some information from the printout.

```
\PL@accent@def{a}{\k a}
\PL@accent@def{c}{\@tabacckludge'c}
\PL@accent@def{e}{\k e}
\PL@accent@def{l}{\PlPrIeC{\l}}
\PL@accent@def{n}{\@tabacckludge'n}
\PL@accent@def{o}{\@tabacckludge'o}
```

```
\PL@accent@def{s}{\@tabacckludge's}
\PL@accent@def{x}{\@tabacckludge'z}
\PL@accent@def{z}{\.z}
\PL@accent@def{A}{\k A}
\PL@accent@def{C}{\@tabacckludge'C}
\PL@accent@def{E}{\k E}
\PL@accent@def{L}{\PlPrIeC{\L}}
\PL@accent@def{N}{\@tabacckludge'N}
\PL@accent@def{O}{\@tabacckludge'O}
\PL@accent@def{S}{\@tabacckludge'S}
\PL@accent@def{X}{\@tabacckludge'Z}
\PL@accent@def{Z}{\.Z}
\PL@accent@def{<}{\PlPrIeC{\guillemotleft}}
\PL@accent@def{>}{\PlPrIeC{\guillemotright}}
\PL@accent@def{,}{\PlPrIeC{\quotedblbase}}
\PL@accent@def{'}{\PlPrIeC{\textquotedblright}}
\PL@accent@def{-}{\PlPrIeC{\dywiz}}
%
\let \PL@accent@def \undefined
```

`\prefixing` The last touch is the definition of the `\prefixing` macro which activates the slash, but only if `plprefix` package was't loaded before. We manage prefixing flag `\pr@fix` for compatibility with MEX.

```
\@ifpackageloaded{plprefix}{}{%
    \def\prefixing{\catcode`/=\active
        \bgroup \uccode`\~=`/ \uppercase{\egroup \let~\PLSlash}%
        \let\pr@fix=T}
```

`\nonprefixing` and `\nonprefixing` macro which deactivates the slash.

```
    \def\nonprefixing{\catcode`/=12 \let\pr@fix=F}
}
```

2.5 Maths in Polish

The next few macros are provided to typeset maths in Polish.

`\arc` In Polish, transcendental functions are written with a tiny space after `arc` or `ar`. Here we define macro `\arc` which when followed by eg. `\sin` typesets arc sin.

```
\def\arc#1{\mathop{\operator@font
     arc\thinspace\escapechar-1 \string#1}\nolimits}
\def\ar#1{\mathop{\operator@font
     ar\thinspace\escapechar-1 \string#1}\nolimits}
```

`\tg` We also use different abbreviations for tangent and cotangent.

`\tgh` `\ctg` `\ctgh`

```
\def\tg{\mathop{\operator@font tg}\nolimits}
\def\ctg{\mathop{\operator@font ctg}\nolimits}
\def\tgh{\mathop{\operator@font tgh}\nolimits}
\def\ctgh{\mathop{\operator@font ctgh}\nolimits}
\def\nwd{\mathop{\operator@font nwd}}
```

Finally we take a drastic step and redefine LATEX's definitions of mathematical functions. This will allow us to keep the markup independent of the language in

which the document is typeset. We think that this is very important, because TEX is today much more than just a typesetting tool, it is also a language which is used to exchange mathematical formulæ. Redefinition will be suppressed when option `nomathsnames` is used.

```
\def\PL@redef@funcnames{%
  \let\tan=\tg     \let\cot=\ctg
  \let\tanh=\tgh   \let\coth=\ctgh
  \def\arcsin{\arc\sin}
  \def\arccos{\arc\cos}
  \def\arctan{\arc\tg}
  \def\arccot{\arc\ctg}
  \let\gcd\nwd
  }
```

These redefinitions should be supplemented by appropriate greater-than-or-equal and less-than-or-equal symbols. They are introduced by the `plmath` option or autodetection, when we are sure we have those symbols available in our fonts.

2.6 Dashes

`\dywiz` When a Polish compound word is split at the hyphen, it should be typeset with two hyphens: one at the end of line and the second at the beginning of the new line. We provide macro `\dywiz` which gives proper hyphenation of compound words. Kerns before and after `\discretionary` allow both parts of the word to be considered for hyphenation.

```
\def\dywiz{\kern0sp\discretionary{-}{-}{-}\penalty10000\hskip0sp\relax}
```

`\pauza` Polish typographical rules require to put a fixed space of .2em around dashes and forbid breaking a line before a dash.

```
\newcommand*\pauza{\unskip\kern.2em\textemdash\hskip.2em\ignorespaces}
\newcommand*\ppauza{\unskip\kern.2em\textendash\hskip.2em\ignorespaces}
```

2.7 Teaching LaTeX to speak Polish

In early versions of LaTeX there were problems when one wanted to customize predefined texts which were inserted automatically by LaTeX (such as *Bibliography* or *Chapter*). They were all hidden deep in the definitions of sectioning or other commands. Now they are all defined as simple macros which can easily be redefined in language packages. We will do that here.

```
\def\prefacename{Przedmowa}
\def\refname{Literatura}
\def\abstractname{Streszczenie}
\def\bibname{Bibliografia}
\def\chaptername{Rozdzia\PLSlash l} % uppercasing in running head must work
\def\appendixname{Dodatek}
\def\contentsname{Spis tre\'sci}
\def\listfigurename{Spis rysunk\'ow}
\def\listtablename{Spis tabel}
```

```
\def\indexname{Skorowidz}
\def\figurename{Rysunek}
\def\tablename{Tabela}
\def\partname{Cz\k e\'s\'c}
\def\enclname{Za\l\k aczniki}
\def\ccname{Do wiadomo\'sci}
\def\headtoname{Do}
\def\pagename{Strona}
\def\seename{zob.}
\def\proofname{Dow\'od}
```

`\today` Finally we redefine the macro `\today` to print the current date in Polish. In Polish documents in some situations it is more appropriate to use the full word *roku* (meaning *year*) at the end of the date and sometimes it is more natural to use an abbreviation. The macro `\PLdateending` which expands at the end of the date can be easily redefined to suit particular needs.

```
\def\today{\number\day~\ifcase\month\or
  stycznia\or lutego\or marca\or kwietnia\or maja\or czerwca\or
  lipca\or sierpnia\or wrze\'snia\or pa\'zdziernika\or
  listopada\or grudnia\fi \space\number\year \PLdateending}
```

2.8 Macros needed later

This macro redefines all standard maths fonts. Now `pl` maths fonts will be used instead of `cm` maths fonts.

```
\def\PL@setmaths{%
```

We start by leaving sign that we have fonts avaiable to redefine `\ge` and `\le` macros.

```
    \def\PLm@ths{}
```

We redefine math alphabets for both math versions. We don't have to redefine `\mathrm`, `\mathnormal` or `\mathcal` alphabets, as they bound to `operators`, `letters` and `symbols` fonts by default (see `fontdef.dtx`.

We must define OT4 encoding if it is not defined yet.

```
    \@ifundefined{T@OT4}{%
      \input ot4enc.def
    }{}
    \SetMathAlphabet{\mathbf}{normal}{OT4}{cmr}{bx}{n}
    \SetMathAlphabet{\mathsf}{normal}{OT4}{cmss}{m}{n}
    \SetMathAlphabet{\mathit}{normal}{OT4}{cmr}{m}{it}
    \SetMathAlphabet{\mathtt}{normal}{OT4}{cmtt}{m}{n}
```

We set math alphabets for bold version.

```
    \SetMathAlphabet{\mathsf}{bold}{OT4}{cmss}{bx}{n}
    \SetMathAlphabet{\mathit}{bold}{OT4}{cmr}{bx}{it}
```

We redeclare all standard symbol fonts. We change the definition of `\@font@warning` macro to not to scare the user with warning messages on the screen about encoding change.

```
    \bgroup\let\@font@warning\@font@info
    \SetSymbolFont{operators}   {normal}{OT4}{cmr} {m}{n}
    \SetSymbolFont{letters}     {normal}{OML}{plm} {m}{it}
    \SetSymbolFont{symbols}     {normal}{OMS}{plsy}{m}{n}
    \SetSymbolFont{largesymbols}{normal}{OMX}{plex}{m}{n}
    \SetSymbolFont{operators}   {bold}  {OT4}{cmr} {bx}{n}
    \SetSymbolFont{letters}     {bold}  {OML}{plm} {b}{it}
    \SetSymbolFont{symbols}     {bold}  {OMS}{plsy}{b}{n}
    \egroup
```

As we have just reloaded the maths fonts, we have some new symbols available. We redefine greater-than-or-equal and less-than-or-equal signs to conform to Polish typographical conventions. This is by analogy with that which was done in section 2.5. Redefinition will be suppressed when option *nomathsnames* is used.

```
    \DeclareMathSymbol{\xleq}{3}{symbols}{172}
    \DeclareMathSymbol{\xgeq}{3}{symbols}{173}
}
%
\def\PL@redef@relations{
    \let\leq=\xleq
    \let\geq=\xgeq
    \let\le=\leq
    \let\ge=\geq
}
```

2.9 Options

Package POLSKI provides a number of options which customize it to the specific environment or needs. They switch LaTeX to different encodings, provide additional macros, etc.

2.9.1 Option `plmath`

This option redefines all standard maths fonts. Now `pl` maths fonts will be used instead of `cm` maths fonts.

```
\DeclareOption{plmath}{%
    \PL@setmaths
}
```

2.9.2 Option `nomathsymbols`

This option supresses redefinition of standard LaTeX's macros for trigonometric functions and for less-or-equal signs.

```
\DeclareOption{nomathsymbols}{%
    \def\PLn@m@thsn@mes{}
}
```

2.9.3 Option `MeX`

This mode should prepare everything to be *markup* compatible with LaTeX. This includes macron redefinition.

```
% \changes{v1.2.2}{2001/08/31}{Redefinition of macron was too early.  OT4
% may be not known yet.}
\DeclareOption{MeX}{%
  \AtBeginDocument{%
    \@ifundefined{T@OT1}{}{%
    \DeclareTextCommand{\=}{OT1}{\dywiz}%
    \DeclareTextAccent{\macron}{OT1}{22}}%
    \@ifundefined{T@T1}{}{%
    \DeclareTextCommand{\=}{T1}{\dywiz}%
    \DeclareTextAccent{\macron}{T1}{9}}%
    \@ifundefined{T@OT4}{}{%
    \DeclareTextCommand{\=}{OT4}{\dywiz}%
    \DeclareTextAccent{\macron}{OT4}{22}}%
    \@ifundefined{T@QX}{}{%
    \DeclareTextCommand{\=}{QX}{\dywiz}%
    \DeclareTextAccent{\macron}{QX}{9}}%
    }%
    \let\xle\xleq
    \let\xge\xgeq
    \let\polish\l@polish
    \let\english\l@english
    \def\MeX{M\kern-.111em\lower.6ex\hbox{E}\kern-.075emX}
    \DeclareRobustCommand\LaMeX{%   after latex.dtx
        L\kern-.36em
        {\setbox0\hbox{T}%
         \vbox to\ht0{\hbox{%
                             \csname S@\f@size\endcsname
                             \fontsize\sf@size\z@
                             \math@fontsfalse\selectfont
                             A}
                      \vss}%
        }%
        \kern-.15em
        \MeX\@}%
    }
```

2.9.4 Option `T1`

This will select T1 encoding for the document.

```
\DeclareOption{T1}{%
    \@ifundefined{T@T1}{\input{t1enc.def}}{}
    \def\encodingdefault{T1}\fontencoding{T1}%
    \def\PL@ncodingd@fined{}
}
```

2.9.5 Option QX

This will select QX encoding for the document.

```
\DeclareOption{QX}{%
    \@ifundefined{T@QX}{\input{qxenc.def}}{}
    \def\encodingdefault{QX}\fontencoding{QX}%
    \def\PL@ncodingd@fined{}
}
```

2.9.6 Option OT1

This will select OT1 encoding for the document.

```
\DeclareOption{OT1}{%
    \def\encodingdefault{OT1}\fontencoding{OT1}%
    \def\PL@ncodingd@fined{}
}
```

2.9.7 Option OT4

This will select OT4 encoding for the document.

```
\DeclareOption{OT4}{%
    \@ifundefined{T@OT4}{\input{ot4enc.def}}{}%
    \def\encodingdefault{OT4}%
    \fontencoding{OT4}%
    \def\PL@ncodingd@fined{}%
}
```

2.9.8 Options for prefixing in verbatim

These decide if prefixing is active in verbatim:

```
\DeclareOption{prefixinginverb}{%
        \def\PL@prefixinginverb{1}%
}
\DeclareOption{noprefixinginverb}{%
        \def\PL@prefixinginverb{0}%
}
```

2.9.9 Options for date ending in \today

```
\DeclareOption{roku}{%
   \def\PLdateending{\nobreakspace roku}
}
\DeclareOption{r.}{%
   \def\PLdateending{\nobreakspace r.}
}
\DeclareOption{noroku}{%
   \def\PLdateending{}
}
```

2.10 Taking off ...

This is almost the end. We process all the options in the order of their definition and switch to french spacing which is also Polish traditional spacing.

```
\ExecuteOptions{prefixinginverb,noroku}
\ProcessOptions
\frenchspacing
```

We now try to autodetect whether the pl fonts reside on the system. We assume, that if there is `OT4cmr.def` file on the system, there are also fonts installed. The autodetection is suppressed if any encoding was switched by package options, or redefined before the package was loaded.

```
\def\tempa{OT1}
\ifx\tempa\f@encoding
    \@ifundefined{PL@ncodingd@fined}{%
        \IfFileExists{ot4cmr.fd}{%
            \typeout{\space\space\space
                Switching to Polish text encoding and Polish maths fonts.}
            \@ifundefined{T@OT4}{%
                \input ot4enc.def
            }{}%
            \def\encodingdefault{OT4}
            \fontencoding{OT4}\selectfont
            \PL@setmaths
        }{%
        \typeout{\space\space\space
          Can't locate Polish fonts.  Will use default encoding.}
        }%
```

We set a checkpoint to warn the user if she enters the document in OT1 encoding.

```
        \def\@PL@OT@check{%
            \bgroup
                \def\tempa{OT1}\ifx\tempa\cf@encoding
                    \@ifpackageloaded{ot1patch}{}{%
                        \PackageError{polski}{%
Zaczynasz skladac dokument uzywajac oryginalnych\MessageBreak
czcionek TeXa.  Czcionki te nie maja kompletu polskich\MessageBreak
znakow.  W zwiazku z tym LaTeX bedzie zglaszal bledy.\MessageBreak
\MessageBreak
Zainstaluj czcionki z dystrybucji MeXa dostepne\MessageBreak
na ftp://ftp.gust.org.pl, sprobuj uzyc czcionek EC\MessageBreak
dodajac opcje T1 do wywolania pakietu polski'ego\MessageBreak
lub w ostatecznosci uzyj pakietu ot1patch.}{}}%
                \fi
            \egroup
            \let\@PL@OT@check=\undefined}%
        \AtBeginDocument{\@PL@OT@check}%
    }{%
    \let\PL@ncodingd@fined=\undefined
```

```
    }%
\fi
```

Now we can redefine LaTeXnames for some maths functions and relations if it was not suppressed.

```
\@ifundefined{PLn@m@thsn@mes}{
  \PL@redef@funcnames
  \@ifundefined{PLm@ths}{}{\PL@redef@relations}
  }{}
```

If prefixing is not to be active in verb, we have to add slash to `\dospecials`:

```
\if 0\PL@prefixinginverb
  \expandafter\def\expandafter\dospecials\expandafter{\dospecials\do\/}
\fi
```

Cleaning up and undefining some local macros.

```
\let\PLn@m@thn@mes=\undefined
\let\PLm@ths=\undefined
\let\PL@setmaths=\undefined
\let\PL@redef@relations=\undefined
\let\PL@redef@funcnames=\undefined
\let\PL@prefixinginverb=\undefined
⟨/style⟩
```

3 Configuring LaTeX's hyphenation patterns

This section provides code that configures LaTeX kernel to include a selected set of hyphenation patterns. Nowadays it is not used since all distributions provide a Babel-enabled LaTeX format.

This code will go to file `hyphen.cfg` which, if found, will be read by IniTeX during format generation instead of the standard LaTeX hyphenation patterns configuration.

First we have to adjust language allocation counter (`\count19`) since kernel (incorrectly) causes `\newlanguage` to start allocation from 1.

```
⟨*hyphenation⟩
\global\count19=-1
```

The rest of actions is put into a group, so our auxiliary macros will automatically disappear when they are no longer needed. Allocations done by `\newlanguage` are global and so are `\patterns` and `\hyphenation`.

```
\begingroup
```

Here I define a few auxiliary macros needed to process `language.dat`. Every time TeX sees a new name he puts it into his name pool and it is never freed. For that reason I don't want to introduce new names for my auxiliary macros. So my first idea was to use control sequences of length 1 (which are not put into the pool). But this approach is risky since "hyphenation files" commonly contain small pieces of code to adjust their behaviour to format or TeX version used. So I've decided to redefine locally a few of standard LaTeX macros. This causes code

to be less readable, but I'll try to make these names somehow mnemonic. (I've chosen macros which contribute directly to the current page, which means for sure they're not used by hyphenation files.)

`\@stopline` will be used as a sentinel delimiting line end.

```
\def\@stopline{\@stopline}
```

`\line` is main macro processing line read from `language.dat`. The line is passed to `\line` as argument with a space and `\@stopline` appended. `\line` checks if the line starts with `=` (synonym definition) and based on that passes the line to `\leftline` or `\rightline`.

```
\def\line#1#2\@stopline{%
  \ifx=#1%
    \leftline#2\@stopline
  \else
    \rightline#1#2\@stopline
  \fi
  }
```

`\leftline` is called for synonym lines (with `=` removed). Such lines should contain only a name for the synonym. So `\leftline` first checks if there is anything after the name and raises and error.

```
\def\leftline#1 #2\@stopline{%
  \ifx\@stopline#2\@stopline\else
    \errhelp{The line should contain only an equals sign followed by
            the synonym name.}%
    \errmessage{Extra stuff on a synonym line in language.dat:^^J
            =#1 #2}\fi
```

Next check if the language name wasn't already used:

```
  \expandafter\ifx\csname l@#1\endcsname\relax \else
    \errhelp{This probably means your ``language.dat'' contains many
        lines starting with `#1' or `=#1'.  ^^JThe language `#1' will
        be redefined.  This may not be what you want.}%
    \errmessage{Language `#1' already defined}\fi
```

Synonyms make no sense when no real language was defined yet. This is checked next. If `\count19` is -1 an error is raised and no definition takes place.

```
  \ifnum\count19=\m@ne
    \errhelp{You cannot put synonyms before first real
        language definition in language.dat.}
    \errmessage{Cannot define `#1' as a language synonym: no language
        defined yet}%
  \else
```

Finally the real definition takes place: `l@<language>` is defined with `\chardef` to be last allocated language number.

```
    \global\expandafter\chardef\csname l@#1\endcsname\count19
    \wlog{\string\l@#1=\string\language\number\count19}
  \fi
  }
```

`\rightline` processes lines that don't start with `=`. Such lines instruct iniTEX to read one or more hyphenation files.

The line is split on first space, `#1` being language name, `#2` list of file names. Note that there is at least one space in input line since we've put one just before `\@stopline`.

```
\def\rightline#1 #2\@stopline{%
```

First check if the language is already defined. If the language name is new it is allocated.

```
  \expandafter\ifx\csname l@#1\endcsname\relax
    \expandafter\newlanguage\csname l@#1\endcsname
  \else
    \errhelp{This probably means your ``language.dat'' contains many
        lines starting with `#1' or `=#1'.  ^^JThe patterns will be
        merged with the ones already loaded.  This may not be what you
        want.}%
    \errmessage{Language `#1' already defined}%
  \fi
```

Then the language is set as current to begin loading of hyphenation patterns.

```
  \language\csname l@#1\endcsname
```

The language name is added to the list of defined languages kept in `\displaylines`.

```
  \edef\displaylines{\displaylines, #1}%
```

For every language there should be at least one patterns file specified. So if `#2` is empty we raise an error.

```
  \ifx\@stopline#2\@stopline
    \errhelp{Hyphenation will be inhibited in language `#1'.}%
    \errmessage{No pattern files specified for language `#1'}%
```

Now `\centerline` processes list of file names delimited with `\@stopline`.

```
  \else
  \begingroup
  \message{Loading hyphenation patterns for #1.}
  \centerline#2\@stopline
  \endgroup
  \fi
  }
```

Macro `\centerline` calls itself recursively until no file name remains on input line. For each name it tries to load the file. Absence of file is considered to be a fatal error.

```
\def\centerline#1 #2\@stopline{%
  \InputIfFileExists{#1}{}{%
    \errhelp{Your language.dat file says I should load a file named
    `#1'.^^J  Check whether this name is correct and the file is
    installed. ^^JThe format will not be generated.}%
    \errmessage{Fatal error: patterns file #1 not found}%
    \endgroup\endgroup\@@end}
  \ifx\@stopline#2\@stopline\else \centerline#2\@stopline\fi
  }
```

`\addvspace` This macro is used to ensure that the line from language.dat ends with exactly one space character.

```
\def\addvspace #1 \*#2\@stopline{%
  \ifx\@stopline#2\@stopline
    \expandafter\def\expandafter\*\expandafter{\* }%
  \fi
}
```

With these auxiliaries we can start actual processing. First the existence of file `language.dat` is checked and the file is opened.

```
\openin1 = language.dat
\ifeof1
  \errhelp{You should have a file named language.dat on your system.
    This file specifies for what languages hyphenation patterns should
    be loaded and where these are kept.  Without this file the format
    will not be generated.}%
  \errmessage{Fatal error: language.dat not found}%
  \endgroup\@@end
\fi
```

`\displaylines` is initialized in such a way that language list won't contain starting comma:

```
\let\displaylines\@gobble
```

Now lines from `language.dat` are read one by one. `\endlinechar` is set to -1 to avoid a space that may get on the end of input line. But after a line is read `\endlinechar` is reset again since code in patterns files may be fragile to such a condition.

```
\loop
  \endlinechar\m@ne
  \read1 to \*%
  \endlinechar`\^^M
```

Empty lines are skipped and others are passed to `\line` with appended single space and the sentinel. Line is read to `\*`. This is safe since this macro is only used locally here, and normal value of `\*` is of no use for hyphenation files.

```
  \ifx\*\empty
  \else
    \expandafter\addvspace\*\* \*\@stopline
    \expandafter\line\*\@stopline
  \fi
```

Processing takes place until end of `language.dat` is found.

```
  \ifeof1\else
\repeat
\closein1
```

Now another sanity check is made: any reasonable `language.dat` should contain at least one language definition. So we refuse to generate format without any hyphenation patterns.

```
\ifnum\count19=-1
  \errhelp{Your language.dat does not instruct LaTeX to load any
    hyphenation patterns.  Since format with  no hyphenation patterns
    is hardly usable I refuse to generate it.  Check your language.dat
    and try again.}%
  \errmessage{Fatal error: No languages defined in language.dat}%
  \endgroup\@@end
\fi
```

Then code to display list of loaded languages is added to `\everyjob` and the group ends.

```
\edef\displaylines{\the\everyjob
  \noexpand\wlog{Loaded hyphenation patterns for\displaylines.}}
\global\everyjob\expandafter{\displaylines}
\endgroup
\language0
\lefthyphenmin=2 \righthyphenmin=3
```
⟨/hyphenation⟩

4 Font encoding OT4

(This section is not needed any more. The definition for OT4 is present in the LaTeX base.)

Here we define a new encoding. Its main purpose is to provide the link between standard accents such as `\'`,`\.` or `\k` (ogonek), and the corresponding characters in the font. Jackowski's fonts will be called `cm`s and `cc`s in this encoding.

⟨*encoding⟩
```
\ProvidesFile{ot4enc.def}[2017/05/04 v1.3.4 Output encoding for polish fonts]
```

Declare the encoding.

```
\DeclareFontEncoding{OT4}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\"}{OT4}{127}
\DeclareTextAccent{\'}{OT4}{19}
\DeclareTextAccent{\.}{OT4}{95}
\DeclareTextAccent{\=}{OT4}{22}
\DeclareTextAccent{\^}{OT4}{94}
\DeclareTextAccent{\`}{OT4}{18}
\DeclareTextAccent{\~}{OT4}{126}
\DeclareTextAccent{\H}{OT4}{125}
\DeclareTextAccent{\u}{OT4}{21}
\DeclareTextAccent{\v}{OT4}{20}
\DeclareTextAccent{\r}{OT4}{23}
```

The ogonek accent is available only under a e A & E. But we have to provide some definition for `\k`. Some accents have to be built by hand as in OT1:

```
\DeclareTextCommand{\k}{OT4}[1]{%
    \TextSymbolUnavailable{\k{#1}}#1}
```

```
\DeclareTextCommand{\b}{OT4}[1]
   {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{29}%
     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}}}
\DeclareTextCommand{\c}{OT4}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
    \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
\DeclareTextCommand{\d}{OT4}[1]
   {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{10}.\hidewidth}}}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AE}{OT4}{29}
\DeclareTextSymbol{\OE}{OT4}{30}
\DeclareTextSymbol{\O}{OT4}{31}
\DeclareTextSymbol{\L}{OT4}{138}
\DeclareTextSymbol{\ae}{OT4}{26}
\DeclareTextSymbol{\guillemotleft}{OT4}{174}
\DeclareTextSymbol{\guillemotright}{OT4}{175}
\DeclareTextSymbol{\i}{OT4}{16}
\DeclareTextSymbol{\j}{OT4}{17}
\DeclareTextSymbol{\l}{OT4}{170}
\DeclareTextSymbol{\o}{OT4}{28}
\DeclareTextSymbol{\oe}{OT4}{27}
\DeclareTextSymbol{\quotedblbase}{OT4}{255}
\DeclareTextSymbol{\ss}{OT4}{25}
\DeclareTextSymbol{\textemdash}{OT4}{124}
\DeclareTextSymbol{\textendash}{OT4}{123}
\DeclareTextSymbol{\textexclamdown}{OT4}{60}
%\DeclareTextSymbol{\texthyphenchar}{OT4}{`\-}
%\DeclareTextSymbol{\texthyphen}{OT4}{`\-}
\DeclareTextSymbol{\textquestiondown}{OT4}{62}
\DeclareTextSymbol{\textquotedblleft}{OT4}{92}
\DeclareTextSymbol{\textquotedblright}{OT4}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT4}{`\`}
\DeclareTextSymbol{\textquoteright}{OT4}{`\'}
```

Some symbols are faked from others:

```
\DeclareTextCommand{\aa}{OT4}
   {{\accent23a}}
\DeclareTextCommand{\AA}{OT4}
   {\leavevmode\setbox0\hbox{h}\dimen@\ht0\advance\dimen@-1ex%
    \rlap{\raise.67\dimen@\hbox{\char'27}}A}
\DeclareTextCommand{\SS}{OT4}
   {SS}
```

In the OT4 encoding, £ and $ share a slot.

```
\DeclareTextCommand{\textdollar}{OT4}{\nfss@text{%
   \ifdim \fontdimen\@ne\font >\z@
      \slshape
   \else
      \upshape
   \fi
```

```
   \char`\$}}
\DeclareTextCommand{\textsterling}{OT4}{\nfss@text{%
   \ifdim \fontdimen\@ne\font >\z@
      \itshape
   \else
      \fontshape{ui}\selectfont
   \fi
   \char`\$}}
```

Declare the composites.

```
\DeclareTextComposite{\k}{OT4}{A}{129}
\DeclareTextComposite{\'}{OT4}{C}{130}
\DeclareTextComposite{\k}{OT4}{E}{134}
\DeclareTextComposite{\'}{OT4}{N}{139}
\DeclareTextComposite{\'}{OT4}{S}{145}
\DeclareTextComposite{\'}{OT4}{Z}{153}
\DeclareTextComposite{\.}{OT4}{Z}{155}
\DeclareTextComposite{\k}{OT4}{a}{161}
\DeclareTextComposite{\'}{OT4}{c}{162}
\DeclareTextComposite{\k}{OT4}{e}{166}
\DeclareTextComposite{\'}{OT4}{n}{171}
\DeclareTextComposite{\'}{OT4}{s}{177}
\DeclareTextComposite{\'}{OT4}{z}{185}
\DeclareTextComposite{\.}{OT4}{z}{187}
\DeclareTextComposite{\'}{OT4}{O}{211}
\DeclareTextComposite{\'}{OT4}{o}{243}
```

⟨/encoding⟩