Typesetting multilingual documents with ANTOMEGA *

Alexej Kryukov

March 20, 2003

Abstract

Antomega is language support package for Lambda, based on the original `omega.sty` file. However, it provides some additional functionality.

1 Introduction

Moving from LaTeX to Ω is always difficult for an average user, since the Ω distribution doesn't include any language support package which could be used as a Babel replacement. The `omega.sty` file, version of 1999/06/01, was released by the Ω developers as a first attempt to make something like 'Omega-Babel', but, unfortunately, this work was not finished. Moreover, more recent versions of `omega.sty` are suitable only for testing some right-to-left languages, but not for regular work. So I prepared my own package, based on the original `omega.sty`, which fixes some bugs and provides some additional functionality.

2 Installation instructions

First, download and install the Ω binaries or ensure that your TeX installation already includes them. Unpack the archive file with ANTOMEGA and move all files to the appropriate directories (for example, everything in `/omega/lambda` to `$$texmf/omega/lambda`, everything in `/omega/ocp` to `$$texmf/omega/ocp`, and so on. If you already have a file named `language.dat` in `$$texmf/omega/lambda/base`, replace it with the provided file in case you want to get correct hyphenation for Russian and/or Greek.

Note that ANTOMEGA still needs some files from the original Ω distribution. The most important files are `ut1enc.def` and `ut1omlgc.fd`. Unfortunately, these files were not included to the most recent Ω distribution. I can't neither include them to my package as is (this might cause name clashes) nor rename them (since I can't rename the default font and the default encoding vector used in Ω). So in case you haven't these files already installed you have to install them separately. Either take them from an older TeX distribution or from the Ω CVS tree.

There are also some additional translation processes (useful mainly for typesetting polytonic (classical) Greek), which you may want to take from old Ω.

*This file has version number 0.5, last revised on 18 Mar 2003.

Of course, after installing new files you have to update the TEX file names database. Don't forget also to rebuild the `lambda` format file (on teTeX or fpTeX systems you have to run

```
fmtutil --byfmt lambda
```

Now you should be able to typeset you documents with ANTOMEGA.

3 Loading ANTOMEGA

One of the main advantages of `omega.sty` was using different commands for setting the main language of the document and for loading additional languages. ANTOMEGA preserves this feature, using the same `\background` and `\load` commands. So if you want to prepare an English document including some Greek text, you can do it by the same way as with `omega.sty`, for example:

```
\usepackage{antomega}
\background{english}
\load{greek}
```

However, `omega.sty` needs two different files for each language: first of them (with the *.bgd extension) is used by the `\background` command, and second (with the *.lay extension) by the `\load` command. Of course, these two files usually have very similar code. ANTOMEGA fixes this problem: both `\background` and `\load` commands load the same language definition file with the .ldf extension, but process it in a different way.

4 Typesetting in different languages

`omega.sty` supported only a limited set of languages, which included `usenglish`, `french` and `greek`. ANTOMEGA supports the same languages, but a separate support for `usenglish` is no longer available. Instead you can load `english` with options `dialect=british` or `dialect=american`, for example:

```
\background[dialect=american]{english}
```

I also added the support file for Russian. Generally speaking, it is not difficult to provide support for a new language, since language definition files are quite independent from the core package, and so you can write a file with definitions for your language without changing anything in `antomega.sty`, using the existing .ldf files as an example.

In the original `omega` package we could use for switching to another language either an environment with the same name as a name of your language, or (for small pieces of text) the `\local<$language>` macro, there `<$language>` is your language name. These commands had to be defined in the language definition file. For example, `usenglish.bgd` defined the `usenglish` environment and the `\localusenglish` command.

These commands are still supported in ANTOMEGA. However, beginning from the version 0.6 ANTOMEGA provides new language switching commands, compatible with the Babel package. So you can use the `\selectlanguage` and `\foreignlanguage` macros and the `otherlanguage` environment exactly as you did with Babel. This means that your old documents may be transferred to Ω with minimal changes.

5 Loading languages with options

As well as the original `omega.sty` file, ANTOMEGA requires the `keyval` package. So all commands used for loading languages may be executed with different parameters, which may take different values. Each language has its own set of such parameters. However, some options are suitable for all supported languages. The most important of them are `input` and `output` parameters, which replace the `inputenc` and `fontenc` packages, used in standard LaTeX.

5.1 The "input" parameter

Of course, this parameter is language-specific. However, there are two values, which are always supported: `utf-8` and `ucs-2`. The later really means "no conversion", since ucs-2 is the native format for Ω. For example, if you want to type an English document with some international symbols encoded in utf-8, you need the following line in your LaTeX preamble:

```
\background[input=utf-8]{english}
```

5.2 The "output" parameter

For this parameter you can use one of the following values: `unicode`, `omega` and `tex`. `unicode` is used by default. Note that the `omlgc` font, distributed with Ω, is not fully compatible with Unicode. For example, it has a specific encoding for the Latin ligatures and the general punctuation. So you have to set `output=omega` if you want to use this font, and `output=unicode` if you have another font, more strictly conforming to the Unicode standard. You my want also set `output=tex` if you prefer using 8-bit fonts in a standard TeX encoding (T1 for Western languages, T2A for Russian, LGR for Greek).

For example, if you want to typeset your English text with the standard EC fonts, but haven't any corresponding font for Greek, you may use the following preamble:

```
\documentclass{article}
\usepackage{antomega}
\background[output=tex]{english}
\load[output=omega]{greek}
```

6 Translation processes

Since the last Ω versions are suitable only for testing purposes, they don't include many useful files, originally provided by J. Plaice ang Y. Haralambous. Partic-

ularly some Ω translation processes were removed, and some are incorrect (e. g. don't correspond to the `omlgc` font. That's why `antomega` provides its own set of `ocp` and `otp` files, which makes it rather independent from Ω's texmf part.

For conversion to different encodings I added some new .otp and .ocp files, which (I hope) work correctly. Version 0.6 includes also some improved translation processes for conversion from commonly used Cyrillic codepages to Unicode. However, some original `.ocp` files are still necessary for `antomega` to work. There also some rarely used (but still supported in `antomega`) files, not present neither in `antomega` nor in the most recent Ω distributions. If you need them, obtain an older Ω and take from there.

7 Selecting fonts with ANTOMEGA

Of course, it is not enough to set an input encoding for your language. You will need also a correct font matching your encoding. With ANTOMEGA you can select a font separately for each script you use. For example, it defines new commands `\westernrm`, `\westernsf` and `\westerntt`. So, if you want to use Computer Modern for English but prefer to keep standard omlgc for Greek, simply put the following line in your preamble:

```
\renewcommand{\westernrm}{cmr}
```

Of course, you can write your own special packages to make selecting a new font a bit more easy. You can even use standard font selecting packages, but, in this case, you must load them *after* ANTOMEGA itself and *before* any language-specific commands. For example:

```
\usepackage{antomega}
\usepackage{palatino}
\background{english}
```

8 The ANTOMEGA code

8.1 Beginning of package

```
\makeatletter
```

8.2 Handling ΩCP files

`\LoadOCPByName` The macro `\LoadOCPByName` takes two arguments: an OCP file name (without extension) and an Ω command which will be used for loading this file. If the referenced .ocp file doesn't exist in user's system, `id.ocp` will be used instead. So it is possible to proceed with document processing, even if some .ocp files were not found.

```
\def\LoadOCPByName#1#2{\IfFileExists{#2.ocp}{\ocp#1=#2}{
   \PackageWarning{antomega}{#2.ocp not found.
      Identity will be used instead.}{}
   \ocp#1=id}}
```

Now we load some commonly used translation processes, using the macro `\LoadOCPByName`.

```
\ocp\IdOCP=id
\LoadOCPByName{\BasicUtfUni}{uniutf2uni}
\LoadOCPByName{\BasicTexUni}{lat2punct}
\LoadOCPByName{\UniToOmega}{uni2omega}
\LoadOCPByName{\Uppercase}{uppercase}
\LoadOCPByName{\Oldstyle}{oldstyle}
\LoadOCPByName{\LatinUniToTex}{uni2t1}
```

`uni2lig.ocp` is used for setting up Latin ligatures. However, the standard TEX ligature mechanism should be a better choice, since using OCP for this purpose may break hyphenation. So I provide an option for turning this translation process off. Note that `uni2lig.ocp` is designed for pure Unicode fonts and it is never used, if `output` is set to 'omega'. In this case you can't turn off processing ligatures via OCP, since the omlgc font doesn't contain ligatures at all.

```
\DeclareOption{ffi}{\LoadOCPByName{\LatinUniToLig}{uni2lig}}
\DeclareOption{noffi}{\LoadOCPByName{\LatinUniToLig}{id}}
\ExecuteOptions{ffi}
```

Now we can define some standard OCP lists, useful generally for languages with Latin-based scripts.

`\NilOCP` An OCP list filled with empty translation processes.

```
\ocplist\NilOCP=
   \addbeforeocplist 500 \IdOCP
   \addbeforeocplist 1750 \IdOCP
   \addbeforeocplist 2000 \IdOCP
   \addbeforeocplist 3500 \IdOCP
\nullocplist
```

`\BasicLatinOCP` This is ANTOMEGA's default OCP list. It doesn't translate text to any other character set.

```
\ocplist\BasicLatinOCP=
   \addbeforeocplist 500 \IdOCP
   \addbeforeocplist 1750 \BasicTexUni
   \addbeforeocplist 2000 \IdOCP
\nullocplist
```

`\BasicLatinUtfOCP` This OCP list should be used for utf-8 encoded texts.

```
\ocplist\BasicLatinUtfOCP=
   \addbeforeocplist 500 \BasicUtfUni
   \addbeforeocplist 1750 \BasicTexUni
   \addbeforeocplist 2000 \IdOCP
\nullocplist
```

The following OCP lists are used to convert a text to an Ω output.

`\LatinUniOutOCP` Conversion to a Unicode font. The only operation which may be performed here is setting up the Latin ligatures.

```
\ocplist\LatinUniOutOCP=
   \addbeforeocplist 3500 \LatinUniToLig
\nullocplist
```

\LatinOmegaOutOCP Conversion to the default omlgc font. Its encoding differs from Unicode, and so a special conversion routine is required.

```
\ocplist\LatinOmegaOutOCP=
   \addbeforeocplist 3500 \UniToOmega
\nullocplist
```

\LatinTexOutOCP Conversion from Unicode to the T1 encoding.

```
\ocplist\LatinTexOutOCP=
   \addbeforeocplist 3500 \LatinUniToTex
\nullocplist
```

\UppercaseOCP Conversion to uppercase.

```
\ocplist\UppercaseOCP=
   \addbeforeocplist 3000 \Uppercase
\nullocplist
```

\OldstyleOCP This OCP list converts ASCII digits to their oldstyle equivalents. Note that it is not compatible with the `omlgc` font.

```
\ocplist\OldstyleOCP=
   \addbeforeocplist 4000 \Oldstyle
\nullocplist
```

8.3 Setting up the default encoding

\uniencoding It is necessary to declare a special encoding for Omega-specific 2-byte fonts. Ω developers called it `UT1`.

```
\def\uniencoding{UT1}
```

\uniencodingfile Now we make a file name from the encoding name.

```
\edef\uniencodingfile{%
     \lowercase{\def\noexpand\uniencodingfile{\uniencoding enc.def}}}%
\uniencodingfile
```

Unfortunately, the `ut1enc.def` file was not included in some releases of Ω. So we check if this file exists, and either load it, or simply declare the encoding.

```
\InputIfFileExists{\uniencodingfile}{}{%
   \DeclareFontEncoding{\uniencoding}{}{}
   \PackageWarning{antomega}{\uniencodingfile\ not found.
      The \uniencoding\ encoding was defined by antomega.}{}
   }
\def\encodingdefault{\uniencoding}
```

8.4 Font issues

The omlgc font is not perfect, but it is included to all standard TEX distributions. So, it will be used by default.

```
\def\rmdefault{omlgc}
```

`Antomega` stores its default font names in the `\westernrm`, `\westernsf` and `\westerntt` variables, since `\rmdefault`, `\sfdefault` and `\ttdefault` will be redefined each time we switch to a new language.

```
\ifx\westernrm\undefined\let\westernrm=\rmdefault\fi
\ifx\westernsf\undefined\let\westernsf=\sfdefault\fi
\ifx\westerntt\undefined\let\westerntt=\ttdefault\fi
```

The omlgc font repeats some TEXnical symbols in the range 0x80–0x0F, not used in Unicode, to make them easy accessible. Of course, it's a sort of hack, and so you may prefer using these symbols in their standard positions, especially if you don't like the omlgc font. However, the following lines, taken from original `omega.sty`, may be optionally executed.

```
\DeclareOption{omlgcspecials}{
   \def\textasciitilde{^^^^0080}
   \def\#{^^^^0083}
   \def\${^^^^0084}
   \def\%{^^^^0085}
   \def\&{^^^^0086}
   \def\textbraceleft{^^^^008b}
   \def\textbackslash{^^^^008c}
   \def\textbraceright{^^^^008d}
   \def\textasciicircum{^^^^008e}
   \def\textunderscore{^^^^008f}}
```

The following `\catcode` settings are optional, since they are necessary only for using 8-bit fonts. However, they are safe anyway.

```
\DeclareOption{texspecials}{
   \catcode"15=12
   \catcode"16=12
   \catcode"17=12
   \catcode"18=12
   \catcode"19=12
   \catcode"1A=12}
```

The following commands are redefined according to the real character placement in the Unicode.

```
\def\S{^^^^00a7}
\def\P{^^^^00b6}
\def\dag{^^^^2020}
\def\ddag{^^^^2021}
\def\i{^^^^0131}
```

Generally speaking, with Ω we should use translation processes rather than active characters. So I made `textasciitilde` an 'other symbol'.

```
\catcode`\~=12
```

8.5 Handling character codes

We can't get correct hyphenation for our 2-byte characters without setting `\catcode`, `\lccode` and `\uccode` for each of them. The following commands simplify making such definitions.

`\makeletter` This command takes two arguments, the first being an uppercase character and the second a corresponding lowercase character, and sets `\lccode` and `\uccode` for both characters.

```
\def\makeletter#1#2{%
   \ifnum\catcode#2=11\else\catcode#2=12 \fi
   \ifnum\catcode#1=11\else\catcode#1=12 \fi
   \uccode#1=#1 \uccode#2=#1%
   \lccode#1=#2 \lccode#2=#2}
```

\makelcletter This command takes two arguments, the first being an uppercase character and the second a corresponding lowercase character, and sets \lccode and \uccode for the lowercase character.

```
\def\makelcletter#1#2{%
   \ifnum\catcode#2=11\else\catcode#2=12 \fi
   \uccode#2=#1%
   \lccode#2=#2}
```

\makeucletter This command takes two arguments, the first being an uppercase character and the second a corresponding lowercase character, and sets \lccode and \uccode for the uppercase characters.

```
\def\makeucletter#1#2{%
   \ifnum\catcode#1=11\else\catcode#1=12 \fi
   \uccode#1=#1%
   \lccode#1=#2}
```

\makesameletter This command takes two arguments, both of them being uppercase or lowercase characters, and sets \lccode and \uccode for character 1 equal to character 2.

```
\def\makesameletter#1#2{%
   \ifnum\catcode#1=11\else\catcode#1=12 \fi
   \uccode#1=\uccode#2%
   \lccode#1=\lccode#2}
```

8.6 Warnings and error messages

\ant@nocodes This command is used to show a warning message if Ω can't find a file with lccodes/uccodes for the specified Unicode range.

```
\ifx\PackageWarningNoLine\@undefined
  \def\ant@nocodes#1#2#3{%
     \message{No file was found with symbol codes}
     \message{for the #2 range #3.}
     \message{You may proceed, but your #1 texts}}
     \message{probably will not be correctly hyphenated.}}
\else
   \providecommand*{\ant@nocodes}[3]{%
      \PackageWarningNoLine{antomega}%
         {No file was found with symbol codes\MessageBreak
            for the #2 range #3.\MessageBreak
            You may proceed, but your #1 texts\MessageBreak
            probably will not be correctly hyphenated.}}
\fi
```

\ant@nopatterns This macro is based on Babel's \@nopatterns command. I've just changed its name in order to avoid conflicts.

```
\ifx\PackageWarningNoLine\@undefined
  \def\ant@nopatterns#1{%
     \message{No hyphenation patterns were loaded for}
     \message{the language `#1'}
     \message{I will use the patterns loaded for \string\language=0
        instead}}
\else
   \providecommand*{\ant@nopatterns}[1]{%
      \PackageWarningNoLine{antomega}%
```

```
125          {No hyphenation patterns were loaded for\MessageBreak
126             the language ‘#1’\MessageBreak
127             I will use the patterns loaded for \string\language=0
128             instead}}
129 \fi
```

`\ant@nolang` This macro defines the error message which will be displayed if the requested language definition file was not found.

```
130 \ifx\PackageWarningNoLine\@undefined
131   \def\ant@nolang#1{%
132      \message{Couldn’t find file omega-#1.ldf!!}}
133 \else
134    \providecommand*{\ant@nolang}[1]{%
135       \PackageWarningNoLine{antomega}%
136          {Couldn’t find file omega-#1.ldf!!}}
137 \fi
```

8.7 Different corrections for standard LaTeX commands

With Ω we usually have to control all commands which print some strings (for example, to headers/footers or to the table of contents), so that they always apply correct translation processes and correct font to the text they produce. However, modifying these commands may be inconvenient if we have to use some packages which also try to redefine them (such as `hyperref`). If you want to prevent `antomega` from modifying these commands, load it with the `nolocalmarks` and `nolocaltoc` options.

`\local@marks` This command is executed every time we are switching to a new language. It applies all rules specific for this language to the text, which is written to headers/footers.

```
138 \DeclareOption{localmarks}{%
139    \def\local@marks#1{%
140    \def\markboth##1##2{%
141       \begingroup%
142          \let\label\relax \let\index\relax \let\glossary\relax%
143          \unrestored@protected@xdef\@themark%
144          {{\foreignlanguage{#1}{##1}}{\foreignlanguage{#1}{##2}}}%
145          \@temptokena \expandafter{\@themark}%
146          \mark{\the\@temptokena}%
147      \endgroup%
148      \if@nobreak\ifvmode\nobreak\fi\fi}%
149    \def\markright##1{%
150       \begingroup%
151          \let\label\relax \let\index\relax \let\glossary\relax%
152          \expandafter\@markright\@themark{\foreignlanguage{#1}}%
153          \@temptokena \expandafter{\@themark}%
154          \mark{\the\@temptokena}%
155        \endgroup%
156        \if@nobreak\ifvmode\nobreak\fi\fi}%
157    \def\@markright##1##2##3{\@temptokena{##1}%
158        \unrestored@protected@xdef\@themark{{\the\@temptokena}%
159        {{##3}}}}}
160 }
161 \DeclareOption{nolocalmarks}{\def\local@marks#1{}}
```

\local@contents The same as the \local@marks command, but for the table of contents.

```
\DeclareOption{localtoc}{\def\local@contents#1{%
   \def\addcontentsline##1##2##3{%
   \addtocontents{##1}{\protect\contentsline{##2}{%
   \foreignlanguage{#1}{##3}}{\thepage}}}}
}
\DeclareOption{nolocaltoc}{\def\local@contents#1{}}
\ExecuteOptions{localmarks,localtoc}
```

Now we must modify \local@marks to prevent Ω from using incorrect translation processes.

We can use .ocp files for making letters uppercase.

```
\def\uppercase#1{{\pushocplist\UppercaseOCP#1}}
\let\MakeUppercase\uppercase
```

\oldstylenums This command supposes that our text font contains old style numerals and that they are mapped to their places in the Unicode Private Use area as defined in AGL. Don't use it with the omlgc font.

```
\def\oldstylenums#1{{\pushocplist\OldstyleOCP#1}}
```

\oaddto This command was taken from the Babel package and renamed in order to avoid conflicts. It is useful for modifying some language-specific commands, predefined in *.lfd files.

```
\def\oaddto#1#2{%
  \ifx#1\@undefined
    \def#1{#2}%
  \else
    \ifx#1\relax
      \def#1{#2}%
    \else
      {\toks@\expandafter{#1#2}%
        \xdef#1{\the\toks@}}%
    \fi
  \fi
}
```

8.8 Loading languages

Standard commands for loading languages (the core of the antomega package).

\background This command requires one arguments which must be a language name and loads it as the first language for our document.

The optional argument is a set of parameters and their values for the given language.

```
\def\backgroung@language#1{\csname background@#1\endcsname}%
\newcommand{\background}[2][]{%
   \IfFileExists{omega-#2.ldf}%
   {\input{omega-#2.ldf}\setkeys{#2}{#1}%
      \AtBeginDocument{\backgroung@language{#2}}%
   \newenvironment{#2}[1][]{\begin{otherlanguage}[####1]{#2}}%
      {\end{otherlanguage}{#2}}%
   \expandafter\newcommand\csname local#2\endcsname[2][]{%
      \foreignlanguage[####1]{#2}{####2}}}%
   {\ant@nolang{#2}}}
```

\load This command takes one argument which must be a language name and loads it in addition to the first language.

The optional argument is a set of parameters and their values for the given language.

Both \background and \load commands are used to define a \local<$language> command and a <$language> environment. Commands and environments with these names were standard way to switch languages in the original omega package, as well as in antomega until the version 0.6. Now they are defined in terms of standard babel-like commands.

```
\newcommand{\load}[2][]{\IfFileExists{omega-#2.ldf}
   {\input{omega-#2.ldf}\setkeys{#2}{#1}%
   \newenvironment{#2}[1][]{\begin{otherlanguage}[####1]{#2}}%
      {\end{otherlanguage}{#2}}
   \expandafter\newcommand\csname local#2\endcsname[2][]{%
      \foreignlanguage[####1]{#2}{####2}}}
   {\ant@nolang{#2}}}
```

8.9 Default values for language-specific settings

First we define some standard values for the punctuation commands, used by lat2punct.otp. The command names are self-explanative.

```
\def\common@punctuation{%
   \def\LeftDoubleQuotationMark{^^^^201c}%
   \def\RightDoubleQuotationMark{^^^^201d}%
   \def\LeftPointingDoubleAngleQuotationMark{^^^^00ab}%
   \def\RightPointingDoubleAngleQuotationMark{^^^^00bb}%
   \def\GermanLeftDoubleQuotationMark{^^^^201e}%
   \def\GermanRightDoubleQuotationMark{^^^^201c}%
   \def\QuestionMark{?}%
   \def\ExclamationMark{!}%
   \def\InvertedQuestionMark{^^^^00bf}%
   \def\InvertedExclamationMark{^^^^00a1}%
   \def\Semicolon{;}%
   \def\Colon{:}%
   \def\NonBreakingSpace{\leavevmode\nobreak\ }}
```

\common@font The \common@font macro will be used at the beginning of the document and also each time we should return to the default fonts (e. g. before switching to another language).

```
\def\common@font{\normalfont\fontfamily{\westernrm}%
   \fontencoding{\uniencoding}\selectfont%
   \let\rmdefault=\westernrm\let\sfdefault=\westernsf%
   \let\ttdefault=\westerntt\let\encodingdefault=\uniencoding}
```

\common@language This macro is used for enabling default hyphenation patterns.

```
\def\common@language{%
   \protect\language=0%
   \lefthyphenmin=2\righthyphenmin=3}
```

\noextrascurrent \originalOmega The \originalOmega macro is used to switch all settings, which could be modified by the language switching commands, to their default values.

```
\def\noextrascurrent#1{\@ifundefined{noextras@#1}{}%
```

```
    {\csname noextras@#1\endcsname}}
\def\originalOmega{\@ifundefined{languagename}{}%
    {\noextrascurrent{\languagename}}%
    \common@language%
    \common@punctuation%
    \common@font%
    \clearocplists%
    }
\AtBeginDocument{\originalOmega}
```

8.10 Language switching commands

`\foreignlanguage` have to be redefined, so we have to unset it first, if necessary.

```
\@ifundefined{foreignlanguage}{}%
    {\let\foreignlanguage\@undefined}
```

`\foreignlanguage` This macro works exactly as Babel's `\foreignlanguage` command, but it takes 3 arguments. The first (optional) argument allows to set any options, defined in the support file for the given language. The second argument is languages's name itself, and the third — the piece of text, which should be typeset in this language.

```
\newcommand{\foreignlanguage}[3][]{%
    \@ifundefined{inlineextras@#2}{\ant@nolang{#2}}
    {{\def\languagename{#2}\setkeys{#2}{#1}%
    \csname inlineextras@#2\endcsname#3}}
```

`\selectlanguage` have to be redefined too.

```
\@ifundefined{selectlanguage}{}%
    {\let\selectlanguage\@undefined}
```

`\selectlanguage` This macro works exactly as Babel's `\selectlanguage` command, but it takes 2 arguments. The second argument is languages's name itself, and the first (optional) allows to set any options, defined in the support file for the given language.

```
\newcommand{\selectlanguage}[2][]{%
    \originalOmega%
    \@ifundefined{blockextras@#2}{\ant@nolang{#2}}%
    {\def\languagename{#2}\setkeys{#2}{#1}%
    \csname blockextras@#2\endcsname}}
```

We have to redefine the `otherlanguage` environment as well.

```
\@ifundefined{otherlanguage}{}%
    {\let\otherlanguage\@undefined}
\@ifundefined{endotherlanguage}{}%
    {\let\endotherlanguage\@undefined}
```

`otherlanguage` This environment works exactly as Babel's `otherlanguage` environment. The only difference is that is has an optional argument allowing to set any options, defined in the .ldf file.

```
\newenvironment{otherlanguage}[2][]{%
    \@ifundefined{blockextras@#2}{\ant@nolang{#2}}
    {\setkeys{#2}{#1}\csname blockextras@#2\endcsname}}{}
```

8.11 Processing options

```
\ProcessOptions
```

8.12 End of package

```
\makeatother
```