The E–Em–Eir package[*]

© Mogens Lemvig Hansen
mogens@kayju.com

2007/02/19

Abstract

The E–Em–Eir package facilitates writing documents that must be produced in both a male and a female form by providing natural and easily remembered commands (control sequences) to type in place of gender specific words.

`\E` `\Em` `\Eir` `\Eirs`

I had occasion to write a couple of documents that needed to be printed in both a male and a female version. To accomplish that conveniently, I developed this E–Em–Eir package which provides four commands, `\E`, `\Em`, `\Eir`, and `\Eirs`, that expand to personal pronouns depending on gender as follows:

	Male	Female
`\E`	he	she
`\Em`	him	her
`\Eir`	his	her
`\Eirs`	his	hers

Thus, just as 'I' is the first person singular pronoun, regardless of gender, so `\E` stands the third person singular pronoun for both genders: `\E` is the singular of 'they'. Accordingly, `\Eir` and `\Eirs` (pronounced to rhyme with 'their(s)') are the possessives, and `\Em` (rhyming with 'them') stands for either 'him' or 'her'.[1]

For example,

```
The user neglects reading the documentation at \Eir peril
```

expands to "The user neglects reading the documentation at his peril" or "The user neglects reading the documentation at her peril" depending on gender. Note that `\Eir` (and the other pronoun commands) automatically add a space after the pronoun where appropriate.[2]

`\swapgender`

You can change gender (!) mid stream with the `\swapgender` command. Thus

---

[*] This file describes version v1.1, 2007/02/19.

[1] I first encountered the pronouns 'E', 'Em', and 'Eir' in M.D. Spivak's *The Joy of TEX* and I have adapted the description of those pronouns from there.

[2] Curtesy of David Carlisle's xspace package which is required by the E–Em–Eir package and included in most distributions of TEX.

```
\E* loves \swapgender\Em for \Eir body
```

`swapgender`

expands to either “She loves him for his body” or “He loves her for her body.” Note that the starred form of the pronoun commands yields a capitalized pronoun. The `\swapgender` command is a *declaration* like `\large`: its effect lasts until the end of the enclosing block (or until the gender is explicitly changed again). Like the font size commands, you can also use an environment form:

```
\E* always praises \begin{swapgender} \Eir cooking; and \E
\end{swapgender} loves \Em for it.
```

expands to “She always praises his cooking; and he loves her for it.”

`\newwordpair`

You can easily add more commands like `\E` and friends. For example,

```
\newwordpair{\child}{son}{daughter}

You say your \child's name and \E is marked with God's
sign so that God knows \Em as His own.  Thus, the Sign of
the Cross is a promise that God's love can always find
your \child.
```

expands to “You say your daughter’s name and she is marked with God’s sign so that God knows her as His own. Thus, the Sign of the Cross is a promise that God’s love can always find your daughter.” in the female case. You could also write `grand\child` for grandson or granddaughter, etc.

Other extensions come to mind:

```
\newwordpair{\parent}{father}{mother}
\newwordpair{\spouse}{husband}{wife}
\newwordpair{\sibling}{brother}{sister}
\newwordpair{\nogoodnameforthisone}{nephew}{niece}
\newwordpair{\orforthis}{uncle}{aunt}
\newwordpair{\Mx}{Mr}{Ms}
```

`\renewwordpair`

Each command defined by `\newwordpair` adds a space after the word where appropriate, and each has a starred form that capitalizes the word. There is also a `\renewwordpair` command if you should want to re-define a command.

`\ifmale`

The commands defined by `\newwordpair` are intended to supply a single gender-specific word. If a whole phrase needs to be different, you can use the `\ifmale`–`\else`–`\fi` construct:

```
...\E is expected to wear␣\ifmale smoking, patent leather
shoes, and top hat\else full-length evening gown\fi\␣for
the gala.
```

This construct is provided by plain TEX, so avoiding spurious spaces is the user’s responsibility. If you find yourself using this construct often, you should have a look at the optional package and/or the version package.

You set the gender when you load the E–Em–Eir package with `\usepackage` in the preamble:

```
\usepackage[⟨option⟩]{eemeir}
```

`\male`
`\female`
`\askforgender`
`\male[...]`
`\female[...]`

The ⟨*option*⟩ can be one of `male`, `female`, or the default, `ask`. You can also set the gender (at any reasonable place in your document) with one of the commands `\male`, `\female`, or `\askforgender`—and don't forget `\swapgender`.

A document can of course refer to the gender of more than one person. The E–Em–Eir package can therefore keep track of more than one person: Define a new person by giving an optional argument to `\male`, `\female`, or `\askforgender`. Then give the same optional argument to `\E` or any other command defined by `\newwordpair` to use the gender of that person. For example,

```
\female[pastor]

Let your \child face the pastor so \E[pastor] can...
```

expands to "Let your son face the pastor so she can. . . " if the main gender is male and `\child` has been defined as above. Once you have set the gender of a new person with, say, `\female[pastor]`, you can change the gender with `\swapgender[pastor]` (or `\begin{swapgender}[pastor]`. . .) and the `\ifmale` construct for this person becomes `\ifmalepastor`–`\else`–`\fi`.

Summary

`\newwordpair`
`\renewwordpair`

Define new commands or re-define old ones with

```
\newwordpair{⟨cmd⟩}{⟨male⟩}{⟨female⟩}
\renewwordpair{⟨cmd⟩}{⟨male⟩}{⟨female⟩}
```

Then ⟨*cmd*⟩ is used like

```
⟨cmd⟩[⟨person⟩]
⟨cmd⟩*[⟨person⟩]
```

The un-starred version expands to ⟨*male*⟩ or ⟨*female*⟩ depending on the current gender of ⟨*person*⟩ or the main gender in the absence of the optional argument. Similarly, the starred version expands to ⟨*male*⟩ or ⟨*female*⟩ with the first letter capitalized.[3]

`\male`
`\female`
`\askforgender`

You set the ⟨*person*⟩ gender (or the main gender in the absence of the optional argument) with one of the commands

```
\male[⟨person⟩]
\female[⟨person⟩]
\askforgender[⟨person⟩]
```

`\swapgender`
`swapgender`

You can also set the main gender with one of the options `male`, `female`, or `ask` (the default) when you load the E–Em–Eir package. Once a gender has been set, you can change it with

---

[3] If you want more than just the first letter capitalized, enclose the desired letters in braces: If `\newwordpair{\strange}{{ab}cd}{{def}gh}`, then `\strange*` expands to "ABcd" or "DEFgh".

`\swapgender[⟨person⟩]`

or

`\begin{swapgender}[⟨person⟩] ...\end{swapgender}`

`\ifmale[...]` If you have longer passages that are gender specific, you can enclose them in `\ifmale`⟨*person*⟩–`\else`–`\fi` or `\ifmale`–`\else`–`\fi` in case of the main gender. If you use this construct often, either you have misunderstood the purpose of the E–Em–Eir package or I should extend it with some suitable environment for conditional inclusion of blocks of text. Let me know...

Installation

As you must have figured, you generate the documentation for the E–Em–Eir package by running the file `eemeir.dtx` through LaTeX—twice to resolve cross references.[4]

To extract the `.sty` file from `eemeir.dtx`, run `eemeir.ins` through LaTeX. You now have to decide what to do with several files.

- Move the file `eemeir.sty` to some directory where LaTeX can find it; the natural choice would be `(local)texmf/tex/latex/misc`.
- Move the documentation, `eemeir.dvi`, to `(local)texmf/doc/latex/misc`.
- You may discard the source files, `eemeir.dtx` and `eemeir.ins`, or store them in `(local)texmf/source/latex/misc`.

License

This program may be distributed and/or modified under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in `http://www.latex-project.org/lppl.txt` and version 1.3 or later is part of all distributions of LaTeX version 2003/12/01 or later.

This program consists of the files `eemeir.dtx` and `eemeir.ins`.

Implementation

`\@eemeir`
`\@eemeirM`
`\@eemeirF`

We need a few control sequences for temporary storage. Define them first with `\newcommand` to verify that they are not in use.

```
⟨*package⟩
\newcommand{\@eemeir}{\relax}
\newcommand{\@eemeirM}{\relax}
\newcommand{\@eemeirF}{\relax}
```

---

[4] If you want an index, you must run MakeIndex (`makeindex -s gind.ist eemeir`) between the two LaTeX runs.

Then declare the options.

```
5 \DeclareOption{male}  {\def\@eemeir{\male[]}}
6 \DeclareOption{female}{\def\@eemeir{\female[]}}
7 \DeclareOption{ask}   {\def\@eemeir{\askforgender[]}}
8 \ExecuteOptions{ask}
```

No matter how many options are processed, in the end `\@eemeir` will (obviously) contain only one of `\male[]`, `\female[]`, or `\askforgender[]`. On code line 79 we'll call on `\@eemeir` to set the main gender.

```
9 \ProcessOptions
10 \RequirePackage{xspace}
```

`\male` We use `\@bsphack` and `\@esphack` to ensure that in things like `␣\male[...]␣` only one of the spaces is typeset. Below, `\ifmale#1` is `\ifmale` if the optional argument is blank and `\ifmale`⟨*person*⟩ if the optional argument is ⟨*person*⟩. Thus, if `\ifmale#1` is undefined, we define a new person by creating a new `\ifmale...` (and otherwise we `\relax`).

```
11 \newcommand{\male}[1][]{%
12   \@bsphack
13   \@ifundefined{ifmale#1}
14     {\expandafter\newif\csname ifmale#1\endcsname}
15     {\relax}%
```

Then set the switch to `true` by calling `\maletrue` or `\male`⟨*person*⟩`true` depending on the optional argument.

```
16   \csname male#1true\endcsname
17   \@esphack}
```

`\female` `\female` is similar; just set the switch with `\male`⟨*person*⟩`false`.

```
18 \newcommand{\female}[1][]{%
19   \@bsphack
20   \@ifundefined{ifmale#1}
21     {\expandafter\newif\csname ifmale#1\endcsname}
22     {\relax}%
23   \csname male#1false\endcsname
24   \@esphack}
```

`\askforgender` `\askforgender` begins just as `\male` and `\female`.

```
25 \newcommand{\askforgender}[1][]{%
26   \@bsphack
27   \@ifundefined{ifmale#1}
28     {\expandafter\newif\csname ifmale#1\endcsname}
29     {\relax}%
```

Then use the temporary variable `\@eemeir` to help type out a pretty message—and store the answer again in `\@eemeir`.

```
30   \def\@eemeir{#1}%
31   \ifx\@eemeir\@empty\def\@eemeir{main}\fi
32   \typein[\@eemeir]{Specify the \@eemeir\space gender:%
33     \space\space male\space\space or\space\space female}%
```

In order to provide a helpful error message if the user types anything but “male” or “female”, we must carefully compare the answer against the legal ones; otherwise a call to `\csname\@eemeir\endcsname[#1]` would have sufficed.

```
  \def\@eemeirM{male}%
  \def\@eemeirF{female}%
  \ifx\@eemeir\@eemeirM\male[#1]%
  \else\ifx\@eemeir\@eemeirF\female[#1]%
  \else\PackageError{eemeir}
    {I'll ignore that}
    {You should have typed either\space\space male\space\space
      or\space\space female}%
  \fi\fi
  \@esphack
}
```

`\swapgender`
`swapgender`

`\newenvironment{swapgender}` defines the command `\swapgender` to be called at the beginning of the environment (and another command to call at the end). We therefore only have to worry about the environment case. Changing a gender that hasn’t been set is meaningless, so `swapgender` insists that `\ifmale#1` is defined.

```
\newenvironment{swapgender}[1][]{%
  \@bsphack
  \@ifundefined{ifmale#1}
    {\PackageError{eemeir}{unknown gender: #1}
      {You must set the #1 gender with \string\male\space
        (or...) first.}}
```

If it’s defined, swap it.

```
    {\csname ifmale#1\endcsname\csname male#1false\endcsname
     \else\csname male#1true\endcsname\fi}%
  \@esphack}
```

The `\end{swapgender}` only has to close the group (which is done by LaTeX automatically) and use the space hacks to ensure that only one space is typeset in cases like ␣`\end{swapgender}`␣.

```
  {\@bsphack\@Esphack}
```

`\newwordpair`
`\renewwordpair`

`\newwordpair` and `\renewwordpair` differ only in which command they employ to make the definition.

```
\newcommand{\newwordpair}  {\@newwordpair\newcommand}
\newcommand{\renewwordpair}{\@newwordpair\renewcommand}
```

`\@newwordpair`

The real work is done by `\@newwordpair`: This next line of code becomes `\(re)newcommand{`⟨*cmd*⟩`}{\@ifstar`....

```
\newcommand{\@newwordpair}[4]{%
  #1{#2}{\@ifstar
```

The `\@ifstar` checks if ⟨*cmd*⟩ is called with or without a star and stores ⟨*male*⟩ and ⟨*female*⟩ in `\@eemeirM` and `\@eemeirF`, respectively. Then `\@eemeirword` is called to check for optional arguments and finish the job.

```
59      {\def\@eemeirM{\MakeUppercase#3}%
60       \def\@eemeirF{\MakeUppercase#4}%
61       \@eemeirword}
62      {\def\@eemeirM{#3}%
63       \def\@eemeirF{#4}%
64       \@eemeirword}}%
65 }
```

`\@eemeirword`

As promised, `\@eemeirword` picks up the optional argument to ⟨*cmd*⟩. If `\ifmale`⟨*person*⟩ is undefined, it's because the user used ⟨*cmd*⟩`[`⟨*person*⟩`]` (something like `\E[pastor]`) before setting the gender of ⟨*person*⟩ with `\male[`⟨*person*⟩`]` or friends. Doing so is really naughty of the user, and maybe I'm being too lenient letting Em get away with it by calling `\askforgender[`⟨*person*⟩`]` for Em.

```
66 \newcommand{\@eemeirword}[1][]{%
67   \@ifundefined{ifmale#1}
68      {\PackageWarning{eemeir}
69        {You should set the #1 gender before using\MessageBreak it}%
70       \askforgender[#1]}
71      {\relax}%
```

If `\ifmale`⟨*person*⟩ is true, set the word stored before in `\@eemeirM`, otherwise use `\@eemeirF`. Then use `\xspace` from the `xspace` package to insert a space if needed.

```
72   \csname ifmale#1\endcsname\@eemeirM\else\@eemeirF\fi
73   \xspace
74 }
```

`\E`
`\Em`
`\Eir`
`\Eirs`

Finally set up the commands that give this package its name

```
75 \newwordpair{\E}{he}{she}
76 \newwordpair{\Em}{him}{her}
77 \newwordpair{\Eir}{his}{her}
78 \newwordpair{\Eirs}{his}{hers}
```

and set the main gender according to options.

```
79 \@eemeir
80 ⟨/package⟩
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.