

P-Lipsum

A ‘lorem ipsum’ paragraph generator
in plain \TeX for plain \TeX ers.

Sergio Spina
sergio.am.spina@gmail.com

May 6, 2013

Contents

I	Introduction	3
II	The program	4
1	Usage.	4
III	Macros	5
2	<code>\lipsum</code>	5
3	<code>\everystartlipsumpar \everyendlipsumpar</code>	5
4	<code>\nopar.</code>	6
IV	Thanks	7
V	The code	8
VI	An example	14
VII	The development tree of the code file	15
VIII	Revision history	16

May 6, 2013

plipsum.nw 2

IX Indexes 17

5 Chunks. 17

6 Identifiers. 17

Part I

Introduction

I wrote this format for my needs, taking inspiration from `lipsum.dtx` by Patrick Happel and `kantlipsum.dtx` by Enrico Gregorio. But those two packages are not useful for me as far as I use only plain \TeX and those packages are written in \LaTeX and aimed only to the \LaTeX world.

Having used *P-Lipsum* for a while and having found it useful I've thought that it can be shared with the \TeX community.

Part II

The program

I used, to build this package, real phrases of real latin, taken from the Cicero's *De finibus bonorum et malorum*. There are two main advantages in using real latin: first, it 'sounds' better; secondly, in the phrases there are a lot of 'fi', 'fl', 'ffi' and 'ffl' groups, useful to see the effects of typographic ligatures with the font choosen for your document.

1 Usage.

To load the format specify `\input plipsum` into your document.

Part III

Macros

2 \lipsum

It's the main macro. It can accept one or two arguments separated by an hyphen and an optional parameter.

The argument is one or two different numbers in the range 1-100.

The optional parameter can be one of the following:

[s]		[short]
[m]		[medium]
[l]		[long]

For ex.:

<code>\lipsum{33}</code>	←	A. one argument
<code>\lipsum[long]{33}</code>	←	B. one optional parameter and one argument
<code>\lipsum{13-33}</code>	←	C. two arguments
<code>\lipsum[s]{13-33}</code>	←	D. one optional parameter and two arguments

In the case 'A': it will be typeset just the paragraph of medium lenght number 33.

In the case 'B': it will be typeset the paragraph of long lenght number 33.

In the case 'C': it will be typeset 21 paragraphs of medium lenght, from the number 13 (included) up to the number 33 (included).

In the case 'D': it will be typeset 21 paragraphs of short lenght, from the number 13 (included) up to the number 33 (included) ¹.

Arguments cannot be greater than 100.

It's ok if the first argument is greater than the second.

`\lipsum{33-13}` ← E. same as example 'C'; it works as well.

3 \everystartlipsumpar \everyendlipsumpar

These two token lists were in existence in the previous version of *P-Lipsum*. As far as they can be easily replaced with normal T_EX programming tools they have been deleted.

¹The interface of the `\lipsum` command is slightly different from the previous version. It is not a good practice to change the interface from one version to another, but *this specific package* is not aimed to typeset documents whose destin is to be archived, so the backward compatibility is not an important issue.

4 `\nopar.`

It eliminates the `\par` between one paragraph and the following one. This way many paragraphs may become one single big paragraph.

`{\nopar\lipsum{14-16}}\par` ← One single paragraph made with all the medium-length paragraphs number 14, number 15 and number 16.

Part IV

Thanks

The ‘lorem ipsum’ paragraphs were generated with the help of

`http://loripsum.net`

maybe the best lipsum generator on the web.

The main source for the paragraphs in lipsum is the Cicero’s *De finibus bonorum et malorum*. You can find the complete, original text at:

`http://www.thelatinlibrary.com/cicero/fin.shtml`

I wrote the package with `noweb` by Norman Ramsey, a simple tool for Literate Programming.

`http://www.cs.tufts.edu/~nr/noweb/`
`http://www.ctan.org/tex-archive/web/noweb`

I feel much more comfortable with this tool than the L^AT_EX packages `Doc` and `Docstrip`.

Part V

The code

Let's begin with the introductory things.

Version informations in a comfortable place.

```
8a  <preliminaries 8a>≡ (15) 8b▷
      \def\PLversion{4}
      \def\PLrevision{2}
      \def\PLrevisiondate{2013/05/06}
```

The sign '@' for the private macros.

```
8b  <preliminaries 8a>+≡ (15) <8a 8c▷
      \chardef\beforeplipsumatcatcode=\the\catcode'@
      \catcode'@=11
```

The first cry of this T_EX child.

```
8c  <preliminaries 8a>+≡ (15) <8b
      {\newlinechar'\message{P-lipsum version
        \PLversion.\PLrevision\space-- revision \PLrevisiondate|}}
```

Let's go with the real thing. The job will be done in two stages:

1. first we will define a collection of "lipsum" paragraphs;
2. then we will define the interfaces (macros) to expand the latin phrases into the document.

Every paragraph is constituted by a macro who's name can be reached at expansion time via a sequential number, for ex.: `\lips@xxv{...}`, `\lips@xxvi{...}`, etc.

Macros from `\lips@i` up to `\lips@c` (the first 100 paragraphs) are long paragraphs; macros from `\lips@ci` up to `\lips@cc` are medium-length paragraphs and the macros from `\lips@cci` up to `\lips@ccc` are short paragraphs.

As far as I'm too lazy to write 300 macro's names organized this way, the job will be done for me by the subsequent macro, thanks to the magic of `\csname ... \endcsname`.

```
8d  <collection 8d>≡ (15) 8e▷
      \newcount\c@parnumber \c@parnumber=0

      \def\create@par{\advance\c@parnumber by1
        \expandafter\def\csname plips@\romannumeral\the\c@parnumber\endcsname}
```

The last touch: I write an equivalent of the primitive `\par` so I can safely 'deactivate' it.

```
8e  <collection 8d>+≡ (15) <8d 9a▷
      \def\@par{\par}
```


It follows the macro to deactivate \@par

9a $\langle collection\ 8d \rangle + \equiv$ (15) $\triangleleft 8e\ 9b \triangleright$
 $\backslash\text{def}\backslash\text{nopar}\{\backslash\text{let}\backslash\text{@par}\backslash\text{space}\}$

Defines:

$\backslash\text{nopar}$, used in chunk 14b.

Now we can define the catalog of latin phrases containing the characteristics builded till now. We won't report all the catalog: it's simply a list of 300 identical macros where the expansion is constituted by a latin paragraph. So it follow just the first paragraph.

9b $\langle collection\ 8d \rangle + \equiv$ (15) $\triangleleft 9a$
 $\% \text{ LONG paragraphs}$
 $\backslash\text{create@par}\{\text{Quid enim necesse est, tamquam meretricem in matronarum coetum, sic voluptatem in virtutum concilium adducere? Nunc dicam de voluptate, nihil scilicet novi, ea tamen, quae te ipsum probaturum esse confidam. Iam quae corporis sunt, ea nec auctoritatem cum animi partibus, comparandam et cognitionem habent faciliorem. Si qua in iis corrigere voluit, deteriora fecit. Polemoni et iam ante Aristoteli ea prima visa sunt, quae paulo ante dixi. Neque solum ea communia, verum etiam paria esse dixerunt. Non enim quaero quid verum, sed quid cuique dicendum sit. Levatio igitur vitiorum magna fit in iis, qui habent ad virtutem progressionis aliquantum. A primo, ut opinor, animantium ortu petitur origo summi boni. Nam si quae sunt aliae, falsum est omnis animi voluptates esse e corporis societate. Quod est, ut dixi, habere ea, quae secundum naturam sint, vel omnia vel plurima et maxima. Praetereo multos, in bis doctum hominem et suavem, Hieronymum, quem iam cur Peripateticum appellem nescio.}\@par\}$

Anyway, after so much work the format launch an echo of proudness in the .log file.

9c $\langle echo\ 9c \rangle \equiv$ (15)
 $\{\backslash\text{newlinechar}'|\backslash\text{message}\{\text{P-lipsum: created}$
 $\backslash\text{number}\backslash\text{c@parnumber}\backslash\text{space paragraphs.}\}\}$

Before of the beginning of the interface building, in which the users will insert values, let's define some error messages.

9d $\langle interface\ 9d \rangle \equiv$ (15) $10a \triangleright$
 $\backslash\text{newhelp}\backslash\text{optparams@error}\{\%$
 $\text{Valid optional parameters are 's', 'short', 'm', 'medium', 'l', 'long'.}\}$
 $\backslash\text{newhelp}\backslash\text{paramexcess@error}\{\%$
 $\text{The best possible value is }\backslash\text{the}\backslash\text{c@parnumber.}\}$

Once builded the foundations we can build the main macro of the format. `\lipsum` has a lot of behaviors:

1. if it is called with a single parameter (specifically a $\langle number \rangle$), it expand to a single medium-length paragraph, namely the one that come at the $(\langle number \rangle + 100)th$ place in the list of paragraphs.
2. If it is called with two parameters $\langle numbers \rangle$ separated by an hyphen it expands to the all the medium-length paragraphs having the number from the lower parameter to the higher (*plus* 100), all the between included.
3. It can be called with an optional parameter, alternatively `s/short` or `m/medium` or `l/long`. If the optional parameter is `s` or `short` the macro expands to the *short* paragraphs having the number from the lower parameter to the higher, all the between included. If the optional parameter is `m` or `medium` the macro expands to the *medium* paragraphs having the number from the lower parameter to the higher, all the between included. So the same for the optional parameters `l` or `long` ($\langle number \rangle + 200$).

So the details of the macro are simple (well, more or less): first the macro look for a square bracket into the parameter:

10a $\langle interface\ 9d \rangle + \equiv$ (15) $\langle 9d\ 10b \rangle$
 `\newif\iflong`
 `\newif\ifmedium`
 `\newif\ifshort`

`\def\lipsum{\futurelet\firstt@k\@lipsum}`

 If a square bracket is found then is called the macro `\opt@par`, otherwise `\nopt@par`. If no square bracket is found it follows that the latin paragraphs have to be medium-length.

10b $\langle interface\ 9d \rangle + \equiv$ (15) $\langle 10a\ 11a \rangle$
 `\def\@lipsum{%`
 `\if[\firstt@k\let\next\opt@par`
 `\else\mediumtrue`
 `\let\next\no@opt@par\fi`
 `\next}`

If a square bracket is found the only thing to do is to set the various boolean values. As the macro `\opt@par` consumes the optional parameter, then it call the servant macro as if it would have been called without optional parameters at all.

```
11a  <interface 9d>+≡ (15) <10b 11b>
      \def\opt@par[#1]#2{\def\param@ne{#1}%
      \def\@l@{l}\def\@long@{long}%
      \def\@m@{m}\def\@medium@{medium}%
      \def\@s@{s}\def\@short@{short}%
      \ifx\param@ne\@l@\shortfalse\mediumfalse\longtrue
      \else
      \ifx\param@ne\@long@\shortfalse\mediumfalse\longtrue
      \else
      \ifx\param@ne\@m@\shortfalse\mediumtrue\longfalse
      \else
      \ifx\param@ne\@medium@\shortfalse\mediumtrue\longfalse
      \else
      \ifx\param@ne\@s@\shorttrue\mediumfalse\longfalse
      \else
      \ifx\param@ne\@short@\shorttrue\mediumfalse\longfalse
      \else
      \errhelp\optparams@error
      {\newlinechar'\errmessage{|P-lipsum: !! ERROR !!
      Wrong optional parameter.}}
      \fi\fi\fi\fi\fi\fi
      \no@opt@par{#2}}
```

Now it is necessary to scan the parameter to ensure that there is an hyphen. In this case is called the macro `\noopt@parA`, otherwise the macro `\noopt@parB`

```
11b  <interface 9d>+≡ (15) <11a 12>
      \newif\ifhyphen

      \def\no@opt@par#1{\scan#1-;\end
      \ifhyphen\noopt@parA#1\end\else\noopt@parB#1\end\fi}
```

`\scan` look for an hyphen into the argument of `\lipsum`. The technique is simple: `\lipsum` calls `\scan` referring to it its own parameter and adding an hyphen and a semicomma. This way `\scan` can be called – accordingly to the parameter of `\lipsum` – alternatively in one of these forms:

1. `\scan<par1>-;\end`
2. `\scan<par1>-<par2>-;\end`

where the hyphens and the `\end` are delimiters of `\scan`. Now the problem is solved: if the second parameter of `\scan` is a semicomma the parameter passed to `\lipsum` has not hyphens and it's a single *<number>*. Otherwise the parameter passed to `\lipsum` contains an hyphen and is thus constituted by two *<numbers>*.

The code is much simpler than the explanation.

12 *<interface 9d>+≡* (15) *<11b 13a>*
`\def\scan#1-#2\end{\ifx;#2\hyphenfalse\else\hyphentrue\fi}`

Now everything is simple and the macros are self-explaining.

If the parameter of `\lipsum` has an hyphen will be performed `\noopt@parA`; otherwise `\noopt@parB`.

13a $\langle interface\ 9d \rangle + \equiv$ (15) $\triangleleft 12$

```

\countdef\c@plipsumAone=253
\countdef\c@plipsumAtwo=251

\def\noopt@parA#1-#2\end{%
  \global\c@plipsumAone=#1\relax
  \global\c@plipsumAtwo=#2\relax
  \ifnum\c@plipsumAone>\c@parnumber
    \warnmsg{first}\fi
  \ifnum\c@plipsumAtwo>\c@parnumber
    \warnmsg{second}\fi
  \ifmedium
    \advance\c@plipsumAone by100\relax
    \advance\c@plipsumAtwo by100\relax
  \else\ifshort
    \advance\c@plipsumAone by200\relax
    \advance\c@plipsumAtwo by200\relax\fi\fi
  \ifnum\c@plipsumAone>\c@plipsumAtwo
    \count@=\c@plipsumAone
    \c@plipsumAone=\c@plipsumAtwo
    \c@plipsumAtwo=\count@\fi
  \types@t}}

\def\noopt@parB#1\end{%
  \ifnum#1>\c@parnumber
    \warnmsg{}\fi
  \count253=#1\relax
  \ifmedium\advance\count253 by100\relax\else
    \ifshort\advance\count253 by200\relax\fi\fi
  \csname plips@\romannumeral\count253\endcsname}

\def\types@t{\let\next\relax
  \ifnum\c@plipsumAone>\c@plipsumAtwo\else
    \csname plips@\romannumeral\the\c@plipsumAone\endcsname
    \advance\c@plipsumAone by1\relax
    \let\next\types@t\fi
  \next}

That's all, folks.

```

13b $\langle ending\ 13b \rangle \equiv$ (15)

```

\catcode'\@=\beforeplipsumatcatcode
\endinput

```

Part VI

An example

The format comes with a simple example (`pliptest.tex`) aimed to show the functionalities of the macros.

```

14a  <banner 14a>≡ (14b 15)
      %% generated with the notangle utility.
      %% The original source file was: plipsum.nw.
      %% Copyright (C) 2013 by Sergio Spina

14b  <pliptest.tex 14b>≡
      %% This is file 'pliptest.tex'
      <banner 14a>
      %%
      %% A few lines of code to show the macros
      %% supplied with the plipsum package.
      \input plipsum
      %~
      One short lipsum paragraph:\par
      \lipsum[s]{13}\vskip\baselineskip
      %~
      One medium-lenght lipsum paragraph:\par
      \lipsum{13}\vskip\baselineskip
      %~
      One long lipsum paragraph:\par
      \lipsum[long]{13}\vskip\baselineskip
      %~
      Three short lipsum paragraphs:\par
      \lipsum[short]{13-15}\vskip\baselineskip
      %~
      Three medium-lenght lipsum paragraphs:\par
      \lipsum[m]{13-15}\vskip\baselineskip
      %~
      Three long lipsum paragraphs:\par
      \lipsum[l]{13-15}\vskip\baselineskip
      %~
      A very long lipsum paragraph:\par
      {\nopar\lipsum[l]{13-31}}\vskip\baselineskip
      %~
      \bye
      %% end of file 'pliptest.tex'
      Uses \nopar 9a.

```

Part VII

The development tree of the code file

```
15  <plipsum.tex 15>≡  
    %% This is file 'plipsum.tex'  
    <banner 14a>  
    <preliminaries 8a>  
    <collection 8d>  
    <paragraphs ??>  
    <echo 9c>  
    <interface 9d>  
    <ending 13b>  
    %% end of file 'plipsum.tex'
```

Part VIII

Revision history

rev. 2.2 2012 aug 27

- published on CTAN

rev. 2.3 2012 sep 14

- rewritten the `\scan` macro, now much more simple and readable.
- added the documentation in noweb.
- solved a bug in `\warnmsg`.

rev. 2.4 2012 nov 23

- added `\everystartplisumpar` and `\everyendplisumpar`.
- replaced all paragraphs.
- many cosmethyc changes.

rev. 3.0 2013 apr 27

- deleted `\everystartplisumpar` and `\everyendplisumpar`.

rev. 4.0 2013 may 05

- modified the interface of `\lipsum`.
- added the optional parameter.
- 300 latin paragraphs.

Part IX

Indexes

5 Chunks.

⟨banner 14a⟩
⟨collection 8d⟩
⟨echo 9c⟩
⟨ending 13b⟩
⟨interface 9d⟩
⟨plipsum.tex 15⟩
⟨pliptest.tex 14b⟩
⟨preliminaries 8a⟩

6 Identifiers.

`\nopar:` 9a, 14b