P-LIPSUM

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

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 I write only plain TEX and they 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.

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 legatures.

Usage.

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

Macros

1. `\lipsum`

It's the main macro. It can accept one or two arguments separated by a comma. For ex.:

`\lipsum 33`	←	A. one argument
`\lipsum 13,33`	←	B. two arguments

In the case 'A': it will be printed just the paragraph number 33.

In the case 'B': there will be printed 21 paragraphs, from the number 13 (included) up to the number 33 (included).

You *must* put a 'space' char after and before the arguments.

`\lipsum64`	←	C. no good. no space before the argument
`\lipsum 28,51\wd0 $\ldots$`	←	D. no good. no space after the arguments
`\lipsum 48 \setbox0 $\ldots$`	←	E. very good.

Arguments cannot be greater than 150.

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

`\lipsum 33,13`	←	F. same as example 'B'; it works as well.

2. `\everystartpar \everyendpar`

These are two token list that are expanded at the beginning and at the end of every paragraph. They can be useful in many tasks that occurs during the writing and the debugging of the TeX code. For ex.:

```
\newcount\counter \counter=1
\everystartpar{{\bf\number\counter.\quad
  \global\advance\counter by1}}
\everyendpar{\space}
\lipsum 69,81
```

`\everystartpar` is expanded just before the first token of the choosen paragraph; `\everyendpar` is expanded just after the last token of the paragraph, before the `\par` between one paragraph and the following one.

3. `\nopar.`

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

	←	One single paragraph made of paragraphs
`{\nopar\lipsum 14,16 }\par`	←	number 14, number 15 and number 16.
	←	Note the space char after the '6'.

Thanks

The 'lorem ipsum' paragraphs were generated with the help of `http://loripsum.net` , maybe the best lipsum generator on the net.

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`

The code

Let's begin with the introductory things.

Version informations in a comfortable place.

⟨*preliminaries*⟩≡
```
\def\version{2} \def\revision{3} \def\revisiondate{2012/09/14}
```

The sign '@' for the private macros.

⟨*preliminaries*⟩+≡
```
\chardef\OldAtCatCode=\the\catcode`@
\catcode`@=11
```

Messages form the format to the `.log` file. It's nice to let them start all on a new line.

⟨*preliminaries*⟩+≡
```
\def\@message#1{{\newlinechar=`|\message{|PLIPSUM.TEX : #1 --- }}}
```

The first cry of this TeX child.

⟨*preliminaries*⟩+≡
```
\@message{Version \version.\revision\space -- revision \revisiondate}
```

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; every paragraph will have it's own accessories and it's own behavior;
2. then we will define the interfaces (macros) to expand the latin phrases from the document.

Every paragraph is equipped with two 'hooks' useful to set something unusual during the expansion. The implementation is constituted by two token registers that are expanded at the beginning and at the ending of every paragraph.

⟨*collection*⟩≡
```
\newtoks\everystartpar \everystartpar={}
\newtoks\everyendpar \everyendpar={}
```

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. But I'm too lazy to write 150 macro's names organized this way, so the subsequent macro will do the job for me thanks to the magic of `\csname ... \endcsname`.

⟨*collection*⟩+≡
```
\newcount\c@parnumber \c@parnumber=0
\def\createpar{\advance\c@parnumber by1
  \expandafter\def\csname plips@\romannumeral\the\c@parnumber\endcsname}
```

The last touch. I overwrite the primitive `\par` so I can safely 'deactivate' it.

⟨*collection*⟩+≡
```
\def\@par{\par}
```

It follows the macro to deactivate `\par`

⟨*collection*⟩+≡

```
\def\nopar{\let\@par\relax}
```

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 150 identical macros where the expansion is constituted by a latin paragraph. So it follow just the first paragraph.

⟨*collection*⟩+≡

```
\createpar{\the\everystartpar Sextilio Rufo, cum is rem ad amicos
ita deferret, se esse heredem. Est autem officium, quod ita factum est,
ut eius facti probabilis ratio reddi possit. Si enim ita est, nisi
contulerimus inter nos, cum finem bonorum dixerimus, quid finis, quid
etiam sit ipsum bonum? Vide ne facinus facias, cum mori suadeas. Intellegi
quidem, ut propter aliam quampiam rem, verbi gratia propter voluptatem,
nos amemus; Itaque contra est, ac dicitis; Expressa vero in iis
aetatibus, quae iam confirmatae sunt. Re mihi non aeque satisfacit, et
quidem locis pluribus. Quod autem magnum dolorem brevem, longinquum levem
esse dicitis, id non intellego quale sit. Dic in quovis conventu te omnia
facere, ne doleas. Apparet statim, quae sint officia, quae actiones.\the
\everyendpar\@par}
```

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

⟨*echo*⟩≡

```
\@message{created \number\c@parnumber\space paragraphs}
```

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

1. if it is called with a single parameter (specifically a `<number>`), it expand to a single paragraph, the one that come at the `<number>`th in the list of paragraphs.
2. if it is called with two parameters `<numbers>` separated by a comma it expands to the all the paragraphs having the number from the lower parameter to the higher, all the between included.

So the details of the macro are simple: first the macro search in the parameter for a comma; if it finds one the typesetting is referred to a servant macro that accept two parameter delimited by a comma; otherwise the job go to another servant macro that accept a single parameter.

⟨*interface*⟩≡

```
\newif\ifcomma
\def\lipsum #1 {\scan#1,;\end
  \ifcomma\lipsumA #1 \else\lipsumB #1 \fi}
```

`\scan` look for a comma into the argument of `\lipsum`. The tecnique is simple: `\lipsum` calls `\scan` referring to it its own parameter and adding a comma 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 commas 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 commas and it's a single `<number>`. Otherwise the parameter passed to `\lipsum` contains a comma and is thus constituted by two numbers.

The code is much simpler than the explanation.

⟨*interface*⟩+≡

```
\def\scan#1,#2\end{\ifx;#2\commafalse\else\commatrue\fi}
```

Before the end some simple macros aimed to error handling.

⟨*interface*⟩+≡

```
\def\sspace{\space\space\space\space\space\space\space\space
  \space\space\space\space\space}

\def\warnmsg#1#2{\@message{ERROR at line \the\inputlineno. The
  #1 parameter of \string\lipsum\space is too big.|
  \sspace You have inserted #2. The best possible value is \the\c@parnumber.|
  \sspace No paragraph will be printed}}

\def\immediatestop{\errmessage{}}
```

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

If the parameter of `\lipsum` has a comma will be performed `\lipsumA`. otherwise `\lipsumB`.

⟨*interface*⟩+≡

```
\countdef\c@plipsumAone=255
\countdef\c@plipsumAtwo=254
\def\lipsumA #1,#2 {{%
  \global\c@plipsumAone=#1\relax
  \global\c@plipsumAtwo=#2\relax
  \ifnum\c@plipsumAone>\c@parnumber
    \@message{\warnmsg{first}{\the\c@plipsumAone}}%
    \immediatestop\fi
  \ifnum\c@plipsumAtwo>\c@parnumber
    \@message{\warnmsg{second}{\the\c@plipsumAtwo}}%
    \immediatestop\fi
  \ifnum\c@plipsumAone>\c@plipsumAtwo
    \count@=\c@plipsumAone
    \c@plipsumAone=\c@plipsumAtwo
    \c@plipsumAtwo=\count@\fi
  \types@t}}

\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}

\def\lipsumB #1 {%
  \ifnum#1>\c@parnumber
    \@message{\warnmsg{}{#1}}%
    \immediatestop\fi
  \csname plips@\romannumeral#1\endcsname}
```

That's all, folks.

⟨*ending*⟩≡

```
\catcode`@=\OldAtCatCode
\endinput
```

An example

The format comes with a simple example aimed to show the functionalities of the macros.

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`.