The file `syntonly.dtx` for use with LaTeX 2ε.[*]
It contains the code for `syntonly.sty`

Frank Mittelbach　　　　Rainer Schöpf

December 10, 2018

This file is maintained by the LaTeX Project team.
Bug reports can be opened (category `latex`) at
https://latex-project.org/bugs.html.

This package implements the `\syntaxonly` declaration for LaTeX 2ε. This command can be used in the preamble for running a document through LaTeX without actually getting any output.

1　Identification

We identify the package and its current version.

```
1 ⟨package⟩\ProvidesPackage{syntonly}
2 ⟨*dtx⟩
3             \ProvidesFile{syntonly.dtx}
4 ⟨/dtx⟩
5 ⟨*package | dtx⟩
6             [2017/06/30 v2.1e Standard LaTeX2e package]
7 ⟨/package | dtx⟩
```

2　Implementation

```
8 ⟨*package⟩
```

`\dummyft@`　First of all we need to define the 'dummy' font.

```
9 \font\dummyft@=dummy \relax
```

`\ifsyntax@`　Now we can define the 'syntax only' feature. We define a switch `\if@syntax` so that any macro can always find out if it is really supposed to typeset text. Its default is to run in normal mode.

```
10 \newif\ifsyntax@
11 \syntax@false
```

`\syntaxonly`　The `\syntaxonly` macro sets up everything for syntax checking.

```
12 \def\syntaxonly{%
```

First of all it sets the `syntax@` switch to `true`.

```
13    \syntax@true
```

Then it globally sets all fonts to the dummy font. These are: the current font outside math mode,

```
14    \global\dummyft@
```

and the 3×16 math fonts for the 16 math *groups*. We use a loop to set these.

```
15    \count@\sixt@@n
16    \loop
17     \ifnum\count@ >\z@
18      \advance\count@\m@ne
19      \global\textfont\count@\dummyft@
20      \global\scriptfont\count@\dummyft@
21      \global\scriptscriptfont\count@\dummyft@
22    \repeat
```

Since all font changes occur either via `\selectfont` (in text or `\mathversion` (for math mode) it is sufficient to change these to no-ops. In addition we must prevent the loading of math fonts, this is done by making `\getanddefine@fonts` a no-op.

```
23    \global\let\selectfont\relax
24    \global\let\mathversion\@gobble
25    \global\let\getanddefine@fonts\@gobbletwo
```

We prevent TEX from complaining about the dummy font having no characters.

```
26    \tracinglostchars\z@
```

Then we disable the output routine, and set `\frenchspacing` (which is slightly faster than `\nonfrenchspacing`). Finally we set `\hbadness` to 10000 to avoid overfull box messages.

```
27    \nopages@
28    \frenchspacing
29    \hbadness\@M}
```

`\nopages@` The `\nopages@` macro disables the LATEX output routine. To this end we define a very simple output routine that empties the output *and* footnote boxes (remember that the latter are insertions.

```
30 \def\nopages@{%
31   \output {\setbox\z@\box\@cclv
32            \setbox\z@\box\footins
33            \deadcycles\z@}%
```

Then we protect it against definition by a style file.

```
34    \newtoks\output
```

But this is not enough: normally the LATEX output routine is responsible for dealing with floating objects. We therefore also redefine the internal macros for handling floats and marginpars.

```
35    \def\@xfloat##1[##2]{%
```

There are a few things that have to be retained: the definition of `\@captype` since it is used by the `\caption` command,

```
36      \def\@captype{##1}%
```

the error message issued when not in outer paragraph mode,

```
37      \ifinner\@parmoderr\fi
```

and the `\@parboxrestore` command for the body of the float. This is necessary since it restores the original definitions of important commands like `\par` or `\\`.

```
38    \setbox\@tempboxa\vbox\bgroup\@parboxrestore}%
```

`\end@float` must now only close the brace:

```
39    \let\end@float\egroup
```

The above would be enough also for two-column floats with the kernel algorithm. However with the refined algorithm inside `fixlxt2e` this doesn't any longer work, so there we also need to explicitly overwrite the end macro for two-column floats (the begin is still okay as it resolves to `\@xfloat` eventually).

```
40    \let\end@dblfloat\egroup
```

The redefinition of the `\marginpar` command is a bit more complicated since we have to check for the optional argument. First we redefine the command itself:

```
41    \def\marginpar{\ifinner\@parmoderr\fi
```

We open a group so that everything gathered in a temporary box can easily be thrown away by closing it again (see below).

```
42       \begingroup \@ifnextchar [\@xmpar\@ympar}
```

`\@xmpar` and `\@ympar` are now defined similar to `\@xfloat` above. If an optional argument is present `\@xmpar` typesets it in a temporary box that is thrown away later. Then it calls up `\@ympar` to process `\marginpar`'s argument.

```
43    \long\def\@xmpar[##1]{%
44       \setbox\@tempboxa\vbox{\@parboxrestore ##1}\@ympar}%
```

`\@ympar` gathers its argument in the same temporary box and throws away its contents by closing the group opened up in `\marginpar` above.

```
45    \long\def\@ympar##1{%
46       \setbox\@tempboxa\vbox{\@parboxrestore ##1}\endgroup}%
```

And that's all we had to do.

```
47 }
```

\@preamblecmds We disable the use of the `\syntaxonly` command after `\begin{document}`

```
48 \@onlypreamble\syntaxonly
49 ⟨/package⟩
```