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

Frank Mittelbach Rainer Schöpf

January 25, 2026

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                [2024/02/08 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{%
```

*This file has version number v2.1e, dated 2024/02/08.

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

```
\syntax@true
```

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

```
\global\dummyft@
```

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

```
\count@\sixt@@n
\loop
 \ifnum\count@ >\z@
  \advance\count@\m@ne
  \global\textfont\count@\dummyft@
  \global\scriptfont\count@\dummyft@
  \global\scriptscriptfont\count@\dummyft@
 \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.

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

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

```
\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.

```
\nopages@
\frenchspacing
\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).

```
\def\nopages@{%
  \output {\setbox\z@\box\@cclv
           \setbox\z@\box\footins
           \deadcycles\z@}%
```

Then we protect it against definition by a style file.

```
\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.

```
\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,

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

the error message issued when not in outer paragraph mode,

```
  \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 \\.

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

\end@float must now only close the brace:

```
\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).

```
\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:

```
\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).

```
    \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.

```
\long\def\@xmpar[##1]{%
  \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.

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

And that's all we had to do.

```
}
```

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

```
\@onlypreamble\syntaxonly
```

⟨/package⟩