The `showlabels` package

Norman Gray
norman@astro.gla.ac.uk

Version 1.4, 2002/04/21

This package helps you keep track of all the labels you define, by putting the name of new labels into the margin whenever the `\label` command is used.

The package allows you to do the same thing for other commands (see below). The only one for which this is *obviously* useful is the `\cite` command, but it is possible to do it also with, for example, the `\ref` or `\begin` commands.

You invoke this package with the command `\usepackage{showlabels}` in the preamble, and you may give the options inner or outer (the default) to have the labels placed in the inner or outer margin of the text. If you don't use the twoside option, then all pages are 'right-hand' pages, and the 'outer margin' is the right hand one.

The package will also work in the presence of the twocolumn option. In this case, the options inner and outer will be ignored, and the label will be placed in the nearer margin.

If you want to change the font the labels appear in, redefine the `\showlabelfont` command. For example, to have labels in a slanted font, you could include the definition `\renewcommand{\showlabelfont}{\small\slfamily}` in the preamble of your document.

If you wish the package to do its magic with the command `\foo` (most typically `\cite`), then give the command `\showlabels{foo}`. The default behaviour of the package is to give the command `\showlabels{label}` internally; if you don't want this to happen – perhaps because you *only* want `\cite` commands highlighted – then give the option nolabel to the `\usepackage` command: `\usepackage[nolabel]{showlabels}`.

By default, the package reprints labels in the margin of the document, but as an alternative, labels can be kept inline as much as possible. To select this, give the option inline, and to select the default behaviour, use marginal.

Compatibility with other packages

The {showlabels} package works by redefining the `\label` command, along with the internal LaTeX commands `\@eqnnum`, `\@makecaption` and `\maketag@@@` (for {amsmath}). All the other commands it defines are 'hidden' by prefixing them with '`SL@`', with the exception of the user command `\showlabelfont`. Each of the three redefined commands carefully includes its previous definition. The {showlabels} package should therefore come *last* of the packages you include using `\usepackage`.

In version 1.1, the package was modified to conform to the slightly different mechanism that `amsmath` uses to produce equation numbers. I don't habitually use `amsmath`, so I won't discover any bugs or weaknesses with its sup-

port here, and I'd consequently be glad to be informed of any that appear. Do note that the `\usepackage{showlabels}` command *must* appear after the `\usepackage{amsmath}` if it is to detect that you are using the {amsmath} package. Note also that, since these additions appeared, AMS-LaTeX and the `amstex` package seem to have been declared 'obsolete' in favour of the `amsmath` package. This package now claims conformance with the `amsmath` package alone, though it will probably work with older versions in fact.

In version 1.3e, the program became compatible with the {hyperref} package in particular, and in general with other packages which themselves modify the `\label` command. This will work, however, only if the {showlabels} package is loaded after other packages which do this. Notwithstanding Sebastian Rahtz's excellent general advice on this, {showlabels} should be loaded after {hyperref}.

For reasonably obvious reasons, this package will not work at all well with the {multicol} package, and for possibly less obvious reasons, it won't work with the leqno option either (at some point it should be modified to at least recognise and warn of the conflict in either case, though it's not obvious to me how to do that). The package *should* now work with {wrapfig}, though I'm not sure that I've exhausted that package's various clevernesses, and there might be some spacing and layout bugs which manifest themselves in that context; using option inline should act as a workaround for any that appear..

The package might still work with LaTeX2.09, but that's neither supported, nor even still tested.

Other packages

I know of, but have not yet played with, the package {showkeys}, by David Carlisle. As far as I know, both packages do roughly the same thing.

History and Credits

Please do let me know if I've omitted anyone from these lists of Helpful People.

For version 1.4, I received comments and bug reports from:

Andrei Shelankov	`<Andrei.Shelankov@tp.umu.se>`
Elmar Walhorn	`<e.walhorn@tu-bs.de>`

For version 1.3, I received comments and bug reports from:

David R. Leal Valmana	`<david@quijote.uc3m.es>`
Sungmo Park	`<smp@dingo.etri.re.kr>`
Olivier Michel	`<Olivier.Michel@lri.fr>`
Jiri Vesely	`<jvesely@karlin.mff.cuni.cz>`
Patrick Sibille	`<sibille@amoco.saclay.cea.fr>`
Hagen Kleinert	`<kleinert@physik.fu-berlin.de>`
Francis M. C. Ching	`<fmcching@kingcong.uwaterloo.ca>`
Michael Friendly	`<friendly@hotspur.psych.yorku.ca>`
lester l. helms	`<l-helms@math.uiuc.edu>`

Many thanks to all.

Back in 29-Jan-92, Darrel Hankerson <hank@ducvax.auburn.edu>, made the update to NFSS, and changed the name to 'showlabel.sty'.

The package was originally released by me on 21-Sep-91, under the name `labels.sty`.

1 Implementation

```
1 ⟨*package⟩
```

Before we do anything else, find out if we're using {amsmath}.... Note that, since these additions appeared, AMS-LaTeX and the `amstex` package seem to have been declared 'obsolete'. This package now claims conformance with the `amsmath` package.

```
2 \newif\ifSL@AMS
3 \expandafter\ifx\csname maketag@@@\endcsname\relax
4   \SL@AMSfalse
5 \else
6   \SL@AMStrue
7   \typeout{with amsmath equation tags}
8 \fi
```

`\@eqnnum` This replacement for `\@eqnnum` will produce a note, sticking into the margin beside the equation number, showing the equation's label. `\SL@labelname` is initialised to `\relax`, redefined within the `\label` macro, and reset to `\relax` here. If it's already equal to `\relax` here, the equation number hasn't been labelled, and so '???' is put in the margin. Don't do this for the {amsmath} case as it artfully incorporates the `\maketag@@@` mechanism into a redefined `\@eqnnum`.

```
9 \ifSL@AMS
```

{amsmath} uses `\maketag@@@`, to form tags in equations. All we have to do is hook into the `\maketag@@@` macro, and use `\df@label`, which is pre-defined with the current label name.

```
10   \let\SL@maketag@@@=\maketag@@@
11   \def\maketag@@@#1{\SL@maketag@@@{#1}%
12   \ifx\df@label\@empty
13 %    \SL@eqnlrtext{???}%
14   \else
15     \SL@eqnlrtext{\df@label}%
16   \fi}
17 \else
```

The 'normal' case, without {amsmath}.

```
18   \let\SL@eqnnum=\@eqnnum
19   \def\@eqnnum{\SL@eqnnum
20   \expandafter\ifx\SL@labelname\relax
21 %    \SL@eqnlrtext{???}%
22   \else
23     \SL@eqnlrtext{\SL@labelname}%
24     \global\def\SL@labelname{\relax}%
25   \fi}
```

And initialise the value of `\labeln@me` to `\relax`, so that `\@eqnnum` starts off behaving the right way.

```
\global\def\SL@labelname{\relax}
\fi
```

Labels are printed with the font specified by `\showlabelfont`, which can be overridden within the document:

```
\def\showlabelfont{\small\tt}
```

Leave this as `\small\tt`, to avoid a pointless incompatibility with LaTeX2.09.

For the benefit of `\SL@prlabelname`, define `\SL@gobblethree` to do nothing other than eat three tokens.

```
\def\SL@gobblethree#1#2#3{}
```

`\SL@prlabelname` Expansion is label name with all catcodes 'other' (Appendix D trickery abounds!). Use `\r@#1`, rather then just `\#1` to avoid defining any new control sequences.

```
\def\SL@prlabelname#1{%
\expandafter\expandafter\expandafter\SL@gobblethree
    \expandafter\string\csname r@#1\endcsname}
```

`\@makecaption` Simple replacement for the `\@makecaption` command. This simply issues the original `\@makecaption` command after redefining `\label`. This does not put the label name in the margin (that's too complicated for the moment), but instead inserts it as part of the caption.

This happens irrespective of anything in `\showlabels`, because this happens through an essentially different mechanism from the way things like `\label` and `\cite` are managed. That's not ideal, but somewhat difficult to work round.

```
\let\SL@makecaption\@makecaption
\long\def\@makecaption#1#2{{\def\label##1{{\showlabelfont
\{\SL@prlabelname{##1}\}\space}\SL@label{##1}}%
```

... and follows it with the original, saved, `\@makecaption`.

```
\SL@makecaption{#1}{#2}}}
```

`\showlabels` Define the `\showlabels` command which allows us to redefine commands which are to have their arguments highlighted. That is, `\showlabels{foo}` means that the command `\foo{bar}` will write 'bar' in the margin, as well as executing whatever `\foo` does normally.

The net result of all this is that a `\showlabels{foo}` command arranges things so that, after `\begin{document}`, `\foo{bar}` expands into `\SL@setlabel{bar}\SL@origfoo{bar}`.

First, define a command `\SL@initfoo`, which, when executed, will save the current (at that time) behaviour of the command `\foo`, and redefine `\foo` to have the required behaviour; we will shortly give this command to `\AtBeginDocument`, so that it is switched on at the correct time, when other packages have done *their* redefinitions of `\foo` (this makes it compatible with the {hyperref} package, which does its own wholesale redefinitions of things like `\label`). Below, the locution `\the\@temptokena` causes the token contents of `\@temptokena` to be included unexpanded in the definition, despite the `\edef`.

```
\def\showlabels#1{%
  \@temptokena=\expandafter{\csname #1\endcsname}
  \expandafter\edef\csname SL@init#1\endcsname{%
    \let\csname SL@orig#1\endcsname\the\@temptokena
```

```
    \let\the\@temptokena\csname SL@#1\endcsname}
  \AtBeginDocument{\csname SL@init#1\endcsname}%
```

Now define `\SL@foo` – it must be undefined when we define `\SL@initfoo` above, so that it isn't expanded in the `\edef`.

```
  \expandafter\def\csname SL@#1\endcsname{\SL@showlabels{#1}}
}
```

`\SL@showlabels` Now we get to the command which does the main processing. The `\SL@showlabels` command calls `\SL@setlabel` to format the label (putting it in the margin, for example), then calls the original `\foo` command (namely `\SL@origfoo`) with the original argument.

The first argument is the name of the command which this command is replacing, for example 'label'. The second is the argument originally given to the command.

We have to be somewhat careful about the positioning of the `\@bsphack` and `\@esphack` commands. The `\@esphack` command should *not* come after the call to `\SL@origfoo`, since that would confuse things terribly if that command took any other arguments, or otherwise messed with the token stream (there's actually no problem in the most common case where we're replacing `\label`, and calling `\SL@origlabel`, but there are problems when we try to replace `\begin` or `\ref` in this way). Instead, adapt the contents of `\@esphack`, but instead of calling `\ignorespaces` when `\@savsk` is positive, add an `\hskip` of 1 scaled point (equal to $1/65536\text{pt} = 5.363 \times 10^{-9}\text{m}$). This will be invisible, but it *is* greater than zero, so that if `\SL@origfoo` itself uses a `\@bsphack`...`\@esphack` pair then the saved `\@savsk` will be positive, and that future `\@esphack` will correctly invoke `\ignorespaces`; that pair will also pick up the `\spacefactor` we restore here. The net result is that the `\SL@setlabel` is invisible, and `\SL@origfoo` is able to make itself invisible, too. Command sequence `\SL@setlabel` is what does the work – this is `\let` equal to either `\SL@margtext` or `\SL@inlinetext` below.

```
\def\SL@showlabels#1#2{%
  \@bsphack
  \SL@setlabel{#2}\relax
  \ifhmode \spacefactor\@savsf \ifdim\@savsk>\z@ \hskip1sp \fi\fi
  \csname SL@orig#1\endcsname{#2}%
}
```

`\SL@margtext` This is the central bit of this package, used by `\SL@showlabels`. The argument is the argument of the `\foo` command which we're processing (for example, the argument to a `\label` command).

Depending on the mode, put the current label name in the margin in one of a variety of ways.

```
\def\SL@margtext#1{%
```

In maths mode, produce a label name alongside the equation number. If we're not using {amsmath}, then save the label name in `\SL@labelname`. If we *are* using {amsmath}, then we don't have to worry, because it's saved in `\df@label` for us.

```
  \ifmmode
    \ifSL@AMS\else
      \xdef\SL@labelname{\SL@prlabelname{#1}}%
    \fi
```

Otherwise, create a box with zero height and depth, and the same width as the page. Put all this in braces, to contain the setting of `\box\@tempboxa` (which probably shouldn't be necessary). The box here we set to be the width of `\hsize`. This is *probably* best, though there's always the worry that `\linewidth` would be the more LaTeX-ish thing to do – using `\columnwidth` is almost certainly wrong, since `\linewidth` can sometimes be changed without `\columnwidth` changing with it (for example, in package {wrapfig}), causing {showlabels} to fail badly.

```
  \else
    \setbox\@tempboxa=\vbox to 0pt{\vss
      \hbox to \hsize{\SL@lrtext{#1}}}%
    \dp\@tempboxa\z@
```

and attach it below the last one, using `\nointerlineskip` if we're in vertical mode, or `\vadjust` otherwise. We need to save and restore the value of `\prevdepth` (which has the sentinel value -1000pt if we're adding this box at the beginning of a vertical list, and `\nointerlineskip` sets `\prevdepth` to this same value). If we don't do this, we get extra ('interline') vertical space added in this case (it might be thought smart to use `\marginpar` here, and so avoid some of this nonsense, but that's not possible since this might be called within boxes, which `\marginpar` objects to).

```
    \ifvmode
      \@tempdima=\prevdepth
      \nointerlineskip\box\@tempboxa\nobreak
      \prevdepth=\@tempdima
    \else
      \vadjust{\box\@tempboxa\nobreak}%
    \fi
```

That's it. Finish off the `\ifmmode`.

```
  \fi
}
```

`\SL@inlinetext` This is an alternative way of formatting the label, which puts it inline as much as possible, and avoids straying into the margins (other than in the case of maths, which is treated as above).

```
\def\SL@inlinetext#1{%
```

Maths mode as with `\SL@margtext`

```
\ifmmode
  \ifSL@AMS\else
    \xdef\SL@labelname{\SL@prlabelname{#1}}%
  \fi
```

In vmode, put the label between lines. Set the box depth to zero to make sure that descenders don't mess up the spacing.

```
\else
  \setbox\@tempboxa=\hbox{\small #1}\dp\@tempboxa\z@
  \ifvmode
    \nointerlineskip\vbox to 0pt{\vss
      \hbox to \columnwidth{\hss \box\@tempboxa}}%
```

And in hmode, squeeze it between the lines, at the current point, carefully taking up no space.

```
  \else
```

```
    \hbox to 0pt{\vbox to 0pt{\vss\box\@tempboxa\vskip 1.5ex}\hss}%
  \fi
\fi
}
```

`\SL@margintext` Set the actual text of the label. Use `\SL@prlabelname` here: without this, a label command given outside of an equation or a `\caption` will appear wrongly if the label has things like underscores within it.

```
\def\SL@margintext#1{{\showlabelfont\{\SL@prlabelname{#1}\}}}
```

But where is the marginal text actually set? It can be in the left margin, the right one, or can alternate. `\SL@lrtext`, used in the `\vbox` above, is set, under the control of `\if@outerlabels` below, to one of `\SL@lefttext`, `\SL@righttext` or `\SL@alternatetext`.

`\SL@righttext`
`\SL@lefttext`

```
\def\SL@righttext#1{\hfill\rlap{\quad\SL@margintext{#1}}}
\def\SL@lefttext #1{\llap{\SL@margintext{#1}\quad}\hfill}
```

The code for `\SL@alternatetext` doesn't work perfectly, as it sometimes manages to get things on the wrong side of the text near the top of a new page. This is a venial slip, however, as this package should never be used in a final version.

`\SL@alternatetext`

```
\def\SL@alternatetext{%
  \if@outerlabels
    \ifodd\c@page
      \let\SL@next\SL@righttext\else
      \let\SL@next\SL@lefttext\fi
  \else
    \ifodd\c@page
      \let\SL@next\SL@lefttext\else
      \let\SL@next\SL@righttext\fi
  \fi
  \SL@next}
```

The case where the twocolumn option is set is slightly different. There we have to switch between placing the note in the left and right margins, depending on whether we're setting the first or second column. This macro, and `\SL@eqntwocoltext` below, uses the switch `\if@firstcolumn` to decide whether it's in the first or the second column of the text (I suppose it'll get terribly confused if we use `multicol.sty` along with this). This is defined and maintained in the base file `ltoutput.dtx`. It's not part of the defined interface, however (there doesn't seem to be one, grump), so I don't suppose we should really rely on it. There isn't an option, however.

`\SL@twocoltext`

```
\def\SL@twocoltext{%
  \if@firstcolumn
    \let\SL@next\SL@lefttext
  \else
    \let\SL@next\SL@righttext
  \fi
  \SL@next}
```

We have very similar things for equations, except that they are set in place, rather than within a zero depth box. This code ASSUMES that equation numbers are going to be on the right hand side of the page. It should probably check for the existence of the leqno option (how?).

\SL@eqnrighttext
\SL@eqnlefttext

```
\def\SL@eqnrighttext#1{\rlap{\quad\SL@margintext{#1}}}
\def\SL@eqnlefttext #1{\hbox to 0pt{\kern -\columnwidth
\llap{\SL@margintext{#1}\quad}\hss}}
```

Now do the analogues for the equation numbers, in the case of the alternate page selection...

\SL@eqnalternatetext

```
\def\SL@eqnalternatetext{%
  \if@outerlabels
    \ifodd\c@page
      \let\SL@next\SL@eqnrighttext\else
      \let\SL@next\SL@eqnlefttext\fi
  \else
    \ifodd\c@page
      \let\SL@next\SL@eqnlefttext\else
      \let\SL@next\SL@eqnrighttext\fi
  \fi
  \SL@next}
```

...and the twocolumn option

\SL@eqntwocoltext

```
\def\SL@eqntwocoltext{%
  \if@firstcolumn
    \let\SL@next\SL@eqnlefttext
  \else
    \let\SL@next\SL@eqnrighttext
  \fi
  \SL@next}
```

To keep track of things, declare the \if@outerlabels switch, and set it true by default.

```
\newif\if@outerlabels
\@outerlabelstrue
```

We select between the various possibilities using the outer and inner options and, implicitly, the twoside option.

```
\DeclareOption{outer}{\@outerlabelstrue}
\DeclareOption{inner}{\@outerlabelsfalse}
```

By default, we run \showlabels{label}. The option nolabel turns this off.

```
\newif\if@showlabellabel
\@showlabellabeltrue
\DeclareOption{nolabel}{\@showlabellabelfalse}
```

\SL@setlabel

Labels can be set either in the margins or inline, by switching between definitions of \SL@setlabel.

```
\DeclareOption{marginal}{\let\SL@setlabel\SL@margtext}
```

```
\DeclareOption{inline}{\let\SL@setlabel\SL@inlinetext}
\let\SL@setlabel\SL@margtext
```

Process any options that have been set.

```
\ProcessOptions
```

and use the values of `\if@outerlabels` and `if@twoside` which may have been set by those options, to set `\SL@lrtext` to be the appropriate control sequence. The presence of the twocolumn option means that we ignore the inner and outer options.

```
\if@twocolumn
  \let\SL@lrtext\SL@twocoltext
  \let\SL@eqnlrtext\SL@eqntwocoltext
\else
  \if@outerlabels
    \if@twoside
  \let\SL@lrtext\SL@alternatetext
  \let\SL@eqnlrtext\SL@eqnalternatetext
    \else
      \let\SL@lrtext\SL@righttext
      \let\SL@eqnlrtext\SL@eqnrighttext
    \fi
  \else
    \if@twoside
  \let\SL@lrtext\SL@alternatetext
  \let\SL@eqnlrtext\SL@eqnalternatetext
    \else
      \let\SL@lrtext\SL@lefttext
      \let\SL@eqnlrtext\SL@eqnlefttext
    \fi
  \fi
\fi
```

Finally, label all the `\label` commands (default, and previous, behaviour), unless this has been suppressed through the nolabel option.

```
\if@showlabellabel
  \showlabels{label}
\fi
```

That's us.

```
⟨/package⟩
```