units.sty – nicefrac.sty

Axel Reichert

1998-08-04

Abstract

units.sty is a package for setting units in a typographically correct way. It is based upon nicefrac.sty, a package for nice fractions. See the files README and COPYING for additional information.

1   Loading

Only nice fractions: \usepackage{nicefrac}
Only units or both: \usepackage{units}

2   Options

Tight spacing for units (default): \usepackage[tight]{units}
Loose spacing for units: \usepackage[loose]{units}
"Nice" fractions (default):   \usepackage[nice]{nicefrac}
"Ugly" fractions: \usepackage[ugly]{nicefrac}

The options nice and ugly can also be used for the units package, they will simply be passed to the nicefrac package, so you can combine the options, e. g.: \usepackage[loose,ugly]{units}

Tight spacing means \, for the space between the value and the dimension, loose spacing uses ~, like 1 m and 1 m. Nice fractions look like ᵐ/ₛ, ugly fractions look like m/s in text mode and $\frac{m}{s}$ in math mode.

3 Commands

Units: \unit[⟨*val*⟩]{⟨*dim*⟩}
Fractions of units: \unitfrac[⟨*val*⟩]{⟨*num*⟩}{⟨*denom*⟩}
Nice fractions: \nicefrac[⟨*fontcmd*⟩]{⟨*num*⟩}{⟨*denom*⟩}

In these list, ⟨*val*⟩ and ⟨*dim*⟩ denote the value and the dimension of the unit, respectively. ⟨*num*⟩ and ⟨*denom*⟩ are the numerator and denominator of the fraction, and ⟨*fontcmd*⟩ can be an author command for fonts or a math alphabet, see fntguide.dvi.

Typically, ⟨*val*⟩ is only a number and ⟨*num*⟩ and ⟨*denom*⟩ are relatively simple LaTeX expressions. If you really feel the need for putting whole paragraphs or complex formula into such a fraction, you have misunderstood the purpose of this package. (-;

A common mistake is to forget the enclosing braces when using the optional argument of the commands inside another optional argument. This is no package bug, but a LaTeX feature. Correct example: \section[{\unit[1]{m}}]{...}.[1]

It is very important to be aware of the fact that all these commands distinguish between text mode and math mode. Within text mode, the font of the surrounding text will be used by default, for math mode \mathrm is the default.

This is quite sensible, because you would not want your collection of delicious recipes (typeset in a mega-cool ultra condensed bold italic calligraphical font) contain those spindle Computer Modern Roman just for half a litre of milk. (-;

Otherwise, when working on scientific papers, strict and consistent notation is really a virtue, and, like it or not, units are typeset with upright fonts. So take great care when deciding about math mode or not.

4 Examples

\sffamily\bfseries\unit{m}	**m**
\sffamily\bfseries$\unit{m}$	m
\sffamily\itshape\unit[1]{m}	*1 m*
\sffamily\itshape$\unit[1]{m}$	1 m
\sffamily\bfseries\unitfrac{m}{s}	**m/s**
\sffamily\bfseries$\unitfrac{m}{s}$	m/s

---

[1]Credits go to FRANK KNAPPE for suggesting this documentation update.

```
\sffamily\itshape\unitfrac[1]{m}{s}        1 m/s
\sffamily\itshape$\unitfrac[1]{m}{s}$      1 m/s
```

As you can see, font changes are ignored in math mode . . .

```
\scriptsize\sffamily\itshape\unitfrac[1]{m}{s}        1 m/s
\scriptsize\sffamily\itshape$\unitfrac[1]{m}{s}$      1 m/s
```

. . . except for the fontsize.

```
\bfseries\itshape\nicefrac{1}{2}        1/2
\bfseries\itshape$\nicefrac{1}{2}$      1/2
\nicefrac[\texttt]{1}{2}                1/2
\nicefrac[\texttt]{\textit{1}}{2}       1/2
$\nicefrac[\mathcal]{A}{B}$             A/B
```

The `\nicefrac` command can deal even with quite strange font changing commands.

5   Typography

Why should units be typeset in upright fonts, not in italics? Because they have to be distinguished from normal variables: "m" is meter, "m" is a variable, for example a mass.

Why should the space between the value and the dimension be non-breakable? Because the reader is disturbed by linebreaks like 1 m.

Why should the space between the value and the dimension be a half word space only? Because things belonging together are typeset tighter. Compare 1 m with 1 m and the normal word spacing, which can vary from line to line.

Why should nice fractions be typeset so that the numerator does not extend above the height of the letter "M" and the denominator does not extend below the baseline? Because otherwise a stretching of the baselineskip could be necessary due to descenders.

By the way, a very common mistake is to place units into brackets, like [N]. The correct notation is $[F] = $ N. The brackets indeed are a function with a variable as an argument. The value returned is the unit. If you need to specify units in table headings, then use a single row for the units, where you put them into *parentheses* instead.

6 Bugs

None, are you kidding?

7 Features

- Consistent and logical markup of units is enhanced instead of fiddling around with spacing and fonts.

- The same command works in text mode *and* math mode, font and size are adjusted automatically for nice integration within text mode while forcing strict notation in math mode.

- Basic requirements of typography are fulfilled: Forbidden linebreaks, correct spacing and the numerator automatically aligning with the height of an "M".

- Easy configuration by use of package options.

8 Bugs again (-;

Ok, ok, you got me.

- Fonts without "M" do not work correctly. Do you know one?

- Fractions in \scriptscriptstyle look ugly because they exceed the height of an "M". As far as I know this is a LaTeX problem, because there are no smaller math fonts available.

- The kerning between numerator and slash or denominator and slash is bad. In fact, there is none.

- The combination of the `ugly` option with text mode can lead to ambiguous fractions. Be happy that a warning is issued. Why do you use this option? Your boss? Ah, I see.

9 Implementation

9.1 Documentation Driver

```
1 ⟨*driver⟩
2 \documentclass[12pt,a4paper]{ltxdoc}
```

```
3 \usepackage{units}
4 \begin{document}
5   \DocInput{units.dtx}
6 \end{document}
7 ⟨/driver⟩
```

9.2   `units.sty`

```
8 ⟨*units⟩
```

9.2.1   Identification

As this package uses the `\RequirePackage` command, it does not work with older LaTeX 2ε versions.

```
9 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
```

The package identifies itself with its release date, a version number, and a short description.

```
10 \ProvidesPackage{units}[1998/08/04 v0.9b Typesetting units]
```

9.2.2   Initialization

The `ifthen` package is loaded because a new boolean variable has to be declared.

```
11 \RequirePackage{ifthen}
12 \newboolean{B@UnitsLoose}
```

9.2.3   Option Declaration

This boolean variable is set according to the package options.

```
13 \DeclareOption{tight}{\setboolean{B@UnitsLoose}{false}}
14 \DeclareOption{loose}{\setboolean{B@UnitsLoose}{true}}
```

Other options will be passed to the `nicefrac` package.

```
15 \DeclareOption*{%
16   \PassOptionsToPackage{\CurrentOption}{nicefrac}%
17 }
```

9.2.4   Option Processing

If no options are specified, tight spacing between the value and the dimension of a unit is used by default. Otherwise the options are processed in the order given by the calling command.

```
18 \ExecuteOptions{tight}
19 \ProcessOptions*
```

9.2.5 Loading Files

Because the macro used for fractions of units is also helpful for other nice fractions, it is build into a stand-alone package called `nicefrac`, which therefore is required by `units.sty`.

```
20 \RequirePackage{nicefrac}[1998/08/04]
```

9.2.6 Defining Commands

In order to make debugging easier for the package user (missing braces or brackets), all commands are declared with the ∗-form of `\DeclareRobustCommand`. The ∗-form does not allow "long" arguments for the commands, so e. g. `\par` commands are forbidden inside the arguments. This should be no serious restriction. (-;

`\unit`   First, it is checked if the optional argument is empty.[2]

```
21 \DeclareRobustCommand*{\unit}[2][]{%
22   \begingroup
23     \def\0{#1}%
24     \expandafter
25   \endgroup
```

If it is, nothing happens.

```
26   \ifx\0\@empty
```

Otherwise it is typeset and followed by a space depending on the boolean declared above.

```
27   \else
28     #1%
29     \ifthenelse{\boolean{B@UnitsLoose}}{~}{\,}%
30   \fi
```

The use of upright fonts for the dimension is forced in math mode. Units in text mode are typeset with the currently active font.

```
31   \ifthenelse{\boolean{mmode}}{\mathrm{#2}}{#2}%
32 }
```

`\unitfrac`   Fractions of units are typeset by means of the `\nicefrac` command. Upright fonts are forced in math mode by use of the optional argument. The rest of code is identical to the former macro.

```
33 \DeclareRobustCommand*{\unitfrac}[3][]{%
34   \begingroup
35     \def\0{#1}%
36     \expandafter
```

---

[2]Credits go to BERND RAICHLE for supplying this method.

```
37   \endgroup
38   \ifx\0\@empty
39   \else
40     #1%
41     \ifthenelse{\boolean{B@UnitsLoose}}{~}{\,}%
42   \fi
43   \ifthenelse{\boolean{mmode}}{%
44     \nicefrac[\mathrm]{#2}{#3}%
45   }%
46   {%
47     \nicefrac{#2}{#3}%
48   }%
49 }
50 ⟨/units⟩
```

9.3   nicefrac.sty

```
51 ⟨∗nicefrac⟩
```

9.3.1   Identification

```
52 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
53 \ProvidesPackage{nicefrac}[1998/08/04 v0.9b Nice fractions]
```

9.3.2   Initialization

First, some new lengths are allocated.

```
54 \newlength{\L@UnitsRaiseDisplaystyle}
55 \newlength{\L@UnitsRaiseTextstyle}
56 \newlength{\L@UnitsRaiseScriptstyle}
```

9.3.3   Loading Files

The ifthen package is loaded for easier handling of decisions.

```
57 \RequirePackage{ifthen}
```

9.3.4   Defining Commands

\@UnitsNiceFrac   Now the code used for nice fractions in math mode:

```
58 \DeclareRobustCommand*{\@UnitsNiceFrac}[3][]{%
59   \ifthenelse{\boolean{mmode}}{%
```

For each of the four different math styles the "M" height is measured, taking specified font changing commands into account.

```
60     \settoheight{\L@UnitsRaiseDisplaystyle}{%
```

```
61        \ensuremath{\displaystyle#1{M}}%
62      }%
63      \settoheight{\L@UnitsRaiseTextstyle}{%
64        \ensuremath{\textstyle#1{M}}%
65      }%
66      \settoheight{\L@UnitsRaiseScriptstyle}{%
67        \ensuremath{\scriptstyle#1{M}}%
68      }%
69      \settoheight{\@tempdima}{%
70        \ensuremath{\scriptscriptstyle#1{M}}%
71      }%
```

The raise height is calculated by taking the difference between the height of a normal "M" and the height of an "M" set in the correct numerator size.

```
72      \addtolength{\L@UnitsRaiseDisplaystyle}{%
73        -\L@UnitsRaiseScriptstyle%
74      }%
75      \addtolength{\L@UnitsRaiseTextstyle}{%
76        -\L@UnitsRaiseScriptstyle%
77      }%
78      \addtolength{\L@UnitsRaiseScriptstyle}{-\@tempdima}%
```

The height of the numerator above the baseline is dependent on the actual fontsize within the math environment.[3] The numerator is put into an appropriately raised box, within math mode (the additional \ensuremath is necessary, because the \raisebox command leaves math mode).

The fontsize for the numerator part is \scriptstyle within \displaystyle and \textstyle, but \scriptscriptstyle within \scriptstyle and \scriptscriptstyle context.

Then the numerator is typeset either with the upright math font or using the font command specified in the optional argument (if it is given). Only the math alphabet commands may be used: \mathnormal, \mathrm, \mathbf, \mathsf, \mathit, \mathtt, and \mathcal.

```
79      \mathchoice
80        {%
81          \raisebox{\L@UnitsRaiseDisplaystyle}{%
82            \ensuremath{\scriptstyle#1{#2}}%
83          }%
84        }%
```

---

[3]Credits go to WOLFGANG MAY for telling me of the \mathchoice command.

```
85        {%
86          \raisebox{\L@UnitsRaiseTextstyle}{%
87            \ensuremath{\scriptstyle#1{#2}}%
88          }%
89        }%
90        {%
91          \raisebox{\L@UnitsRaiseScriptstyle}{%
92            \ensuremath{\scriptscriptstyle#1{#2}}%
93          }%
94        }%
```

Within \scriptscriptstyle context, LaTeX does *not* use even smaller fonts for numerator or denominator. Following the above calculation scheme would lead to a raise height of $0\,\mathrm{pt}$. Because this could make fractions at least ambiguous (if not wrong, like $\mathrm{J/kg \cdot K}$ for specific thermal capacity), the numerator in \scriptscriptstyle context is raised by the same amount as in \scriptstyle context and so exceeds the height of an adjacent "M". This is ugly, but cannot be avoided.

```
95        {%
96          \raisebox{\L@UnitsRaiseScriptstyle}{%
97            \ensuremath{\scriptscriptstyle#1{#2}}%
98          }%
99        }%
```

The slash is typeset with reduced spacing to the numerator and to the denominator.

```
100       \mkern-2mu/\mkern-1mu%
```

The denominator needs a seperate group to delimit the range of the font commands.[4] The choice for the fontsize and the font changing are made just like for the numerator.

```
101       \bgroup
102         \mathchoice
103           {\scriptstyle}%
104           {\scriptstyle}%
105           {\scriptscriptstyle}%
106           {\scriptscriptstyle}%
107         #1{#3}%
108       \egroup
109     }%
```

---

[4]Credits go to MICHAEL SWITKES for the bug report and to HEIKO OBERDIEK for telling me why the fix works.

Now the code for nice fractions in text mode:

```
110   {%
```

First, the height of a normal "M" is measured.

```
111      \settoheight{\L@UnitsRaiseTextstyle}{#1{M}}%
```

LaTeX offers no commands for relative fontsizes. To make the macro work for example within footnotes or chapter headings, it is necessary to avoid absolute sizing commands like `\normalsize` or `\scriptsize`. To obtain a fontsize that suits for the numerator and denominator, some code was taken from the LaTeX kernel (`latex.ltx`, definition of `\textsuperscript`).

In all these measurements the font changing effect of the optional argument is taken into account. Only the following font commands may be used: `\textnormal`, `\textrm`, `\textsf`, `\texttt`, `\textbf`, `\textmd`, `\textup`, `\textit`, `\textsc`, and `\textsl`.

```
112      \settoheight{\@tempdima}{%
113         \ensuremath{%
114            \mbox{\fontsize\sf@size\z@\selectfont#1{M}}%
115         }%
116      }%
```

Both lengths are subtracted. The result is used for the raise of the numerator.

```
117      \addtolength{\L@UnitsRaiseTextstyle}{-\@tempdima}%
```

Finally, the numerator is typeset in "relative scriptsize" either with the normal text font or using the font command specified in the optional argument (if it is given).

```
118      \raisebox{\L@UnitsRaiseTextstyle}{%
119         \ensuremath{%
120            \mbox{\fontsize\sf@size\z@\selectfont#1{#2}}%
121         }%
122      }%
```

The code for the slash and the denominator should now be straight forward.

```
123      \ensuremath{\mkern-2mu}/\ensuremath{\mkern-1mu}%
124      \ensuremath{%
125         \mbox{\fontsize\sf@size\z@\selectfont#1{#3}}%
126      }%
127   }%
128 }
```

`\@UnitsUglyFrac` Ugly fractions in math mode ...

```
129 \DeclareRobustCommand*{\@UnitsUglyFrac}[3][]{%
130   \ifthenelse{\boolean{mmode}}{%
```

... are typeset just like normal fractions.

```
131     \frac{#1{#2}}{#1{#3}}%
132   }%
```

In text mode ...

```
133   {%
```

... they are typeset like J/kg·K.

```
134     #1{#2}/#1{#3}%
```

Because the fraction can get ambiguous or even wrong, a warning message is issued.

```
135     \PackageWarning{nicefrac}{%
136       You used \protect\nicefrac\space or
137       \protect\unitfrac\space in text mode\MessageBreak
138       and specified the ''ugly'' option.\MessageBreak
139       The fraction may be ambiguous or wrong.\MessageBreak
140       Please make sure the denominator is
141       correct.\MessageBreak
142       If it is, you can safely ignore\MessageBreak
143       this warning
144     }%
145   }%
146 }
```

9.3.5   Option Declaration

Dependent on the package options, the `\nicefrac` command typesets either nice or ugly fractions by means of the corresponding macros.

```
147 \DeclareOption{nice}{%
148   \DeclareRobustCommand*{\nicefrac}{\@UnitsNiceFrac}%
149 }
150 \DeclareOption{ugly}{%
151   \DeclareRobustCommand*{\nicefrac}{\@UnitsUglyFrac}%
152 }
```

9.3.6   Option Processing

If no options are specified, nice fractions are used by default. Otherwise the options are processed in the order given by the calling command.

```
153 \ExecuteOptions{nice}
154 \ProcessOptions*
155 ⟨/nicefrac⟩
```

References

[1] GOOSSENS, M.; MITTELBACH, F.; SAMARIN, A.: *Der LATEX-Begleiter*. Addison-Wesley, Bonn, 1st edn., 1994.

[2] KOPKA, H.: *Einführung*, vol. 1 of *LATEX*. Addison-Wesley, Bonn, 1st edn., 1994.

[3] KOPKA, H.: *Ergänzungen – mit einer Einführung in METAFONT*, vol. 2 of *LATEX*. Addison-Wesley, Bonn, 1st edn., 1995.

[4] KOPKA, H.: *Erweiterungen*, vol. 3 of *LATEX*. Addison-Wesley-Longman, Bonn, 1997.

[5] WILLBERG, H. P.; FORSSMAN, F.: *Lesetypographie*. Schmidt, Mainz, 1997.