`url.sty` version 3.4

Donald Arseneau*

2013-09-16

The package defines a form of `\verb` command that allows linebreaks at certain characters or combinations of characters, accepts reconfiguration, and can usually be used in the argument to another command. It is intended for formatting email addresses, hypertext links, directories/paths, etc., which normally have no spaces. The font used may be selected using the `\urlstyle` command, and new url-like commands may be defined using `\urldef`. This package does not make hyper-links! For that purpose, see the hyperref package (or some other deprecated ones).

Usage	Conditions
`\url{ }`	The argument must not contain unbalanced braces. If used in the argument to another command, the `\url` argument cannot contain any "`%`", "`#`", or "`^^`", or end with "`\`".
`\url| |`	where "`|`" is any character not used in the argument and not "`{`" or a space. The same restrictions apply as above except that the argument may contain unbalanced braces.
`\xyz`	for the defined-url "`\xyz`"; such a command can be used anywhere, no matter what characters it contains.

The "`\url`" command is fragile, and its argument is likely to be very fragile, but a defined-url is robust.

1 Package options

Package Option: `obeyspaces`

Ordinarily, all spaces are ignored in the url-text. The "`[obeyspaces]`" option allows spaces, but may introduce spurious spaces when a url containing "`\`" characters is given in the argument to another command. So if you need to obey spaces

*Thanks to Robin Fairbairns for documentation conversion!

you can say "`\usepackage[obeyspaces]{url}`", and if you need both spaces and backslashes, use a defined-url.

Package Option: `hyphens`

Ordinarily, breaks are not allowed after "`-`" characters because this leads to confusion. (Is the "`-`" part of the address or just a hyphen?) The package option "`[hyphens]`" allows breaks after explicit hyphen characters. The `\url` command will **never ever** hyphenate words.

Package Option: `spaces`

Likewise, given the "`[obeyspaces]`" option, breaks are not usually allowed after the spaces, but if you give the options "`[obeyspaces,spaces]`", `\url` will allow breaks at those spaces.

> Note that it seems logical to allow the sole option "`[spaces]`" to let input spaces indicate break points, but not to display them in the output. This would be easy to implement, but is left out to avoid(?) confusion.

Package Option: `lowtilde`

Normal treatment of the `~` character is to use the font's "`\textasciitilde`" character, if it has one (or claims to). Otherwise, the character is faked using a mathematical "`\sim`". The "`[lowtilde]`" option causes a faked character to be used always (and a bit lower than usual).

Package Option: `allowmove`

This option suppresses the test for `\url` being used in a so-called moving argument (check "fragile command"). Using it will enable `\url` to function in more contexts, but when it does fail, the error message may be incomprehensible.

2 Defining a defined-url

Take for example the email address "`myself%node@gateway.net`" which could not be given (using "`\url`" or "`\verb`") in a caption or parbox due to the percent sign. This address can be predefined with

```
\urldef{\myself}\url{myself%node@gateway.net} or
\urldef{\myself}\url|myself%node@gateway.net|
```

and then you may use "`\myself`" instead of "`\url{myself%node@gateway.net}`" in an argument, and even in a moving argument like a caption because a defined-url is robust.

3 Style

You can switch the style of printing using "`\urlstyle{`*xx*`}`", where "*xx*" can be any defined style. The pre-defined styles are "`tt`", "`rm`", "`sf`" and "`same`" which all allow the same linebreaks but use different fonts — the first three select a specific font and the "`same`" style uses the current text font. You can define your own styles with different fonts and/or line-breaking by following the explanations below. The "`\url`" command follows whatever the currently-set style dictates.

4 Alternate commands

It may be desireable to have different things treated differently, each in a predefined style; e.g., if you want directory paths to always be in typewriter and email addresses to be roman, then you would define new url-like commands as follows:

```
\DeclareUrlCommand⟨command⟩{⟨settings⟩}
\DeclareUrlCommand\email{\urlstyle{rm}}
\DeclareUrlCommand\directory{\urlstyle{tt}}.
```

In fact, this `\directory` example is exactly the `\path` definition which might be pre-defined by the package. Furthermore, basic `\url` is defined with

```
\DeclareUrlCommand\url{},
```

without any *settings*, so it uses whatever `\urlstyle` and other settings are already in effect.

You can make a defined-url for these other styles, using the usual `\urldef` command as in this example:

```
\urldef{\myself}{\email}{myself%node.domain@gateway.net}
```

which makes `\myself` act like `\email{myself%node.domain@gateway.net}`, if the `\email` command is defined as above. The `\myself` command would then be robust.

5 Defining styles

Before describing how to customize the printing style, it is best to mention something about the unusual implementation of `\url`. Although the material is textual in nature, and the font specification required is a text-font command, the text is actually typeset in *math* mode. This allows the context-sensitive linebreaking, but

also accounts for the default behavior of ignoring spaces. (Maybe that underlying design will eventually change.) Now on to defining styles.

To change the font or the list of characters that allow linebreaks, you could redefine the commands `\UrlFont`, `\UrlBreaks`, `\UrlSpecials`, etc., directly in the document, but it is better to define a new 'url-style' (following the example of `\url@ttstyle` and `\url@rmstyle`) which defines all of `\UrlBigbreaks`, `\UrlNoBreaks`, `\UrlBreaks`, `\UrlSpecials`, and `\UrlFont`.

5.1 Changing font

The `\UrlFont` command selects the font. The definition of `\UrlFont` done by the pre-defined styles varies to cope with a variety of LaTeX font selection schemes, but it could be as simple as `\def\UrlFont{\tt}`. Depending on the font selected, some characters may need to be defined in the `\UrlSpecials` list because many fonts don't contain all the standard input characters.

5.2 Changing linebreaks

The list of characters after which line-breaks are permitted is given by the two commands (list macros) `\UrlBreaks` and `\UrlBigBreaks`. They consist of repeating `\do\c` for each relevant character `c`.

The differences are that 'BigBreaks' typically have a lower penalty (more easily chosen) and do not break within a repeating sequence (e.g., "`DEC::NODE`"). (For gurus: 'BigBreaks' are treated as mathrels while 'Breaks' are mathbins; see *The TeXbook*, p. 170.) The result is that a series of consecutive 'BigBreak' characters will break at the end and only at the end; a series of 'Break' characters will break after the first and after every following *pair*; there will be no break between a 'Break' character and a following 'BigBreak' char; breaks are permitted when a 'BigBreak' character is followed by 'Break' or any other char. In the case of `http://` it doesn't matter whether `:` is a 'Break' or 'BigBreak' — the breaks are the same in either case; but for (now ancient) *DECnet* addresses using `::` it was important to prevent breaks *between* the colons, and that is why colons are 'BigBreaks'. (The only other 'BigBreak' character is, optionally, the hyphen; slashes are regular 'Break's.)

It is possible for characters to prevent breaks after the next following character (this is used for parentheses). Specify these in `\UrlNoBreaks`.

You can allow some spacing around the breakable characters by assigning

```
\Urlmuskip = 0mu plus 1mu
```

(with `mu` units because of math mode). You can change the penalties used for BigBreaks and Breaks by assigning

```
\mathchardef\UrlBreakPenalty=100
\mathchardef\UrlBigBreakPenalty=100
```

The default penalties are `\binoppenalty` and `\relpenalty`. These have such odd non-LaTeX syntax because I don't expect people to need to change them often. (The `\mathchardef` does not relate to math mode; it is only a way to store a number without consuming registers.)

5.2.1 Arbitrary character actions

You can do arbitrarily complex things with characters by specifying their definition(s) in `\UrlSpecials`. This makes them 'active' in math mode (mathcode `"8000`). The format for setting each special character `c` is: `\do\c{`⟨*definition*⟩`}`, but other definitions not following this style can also be included.

Here is an example to make "`!`" inside `\url` force a line break instead of being treated verbatim (it uses LaTeX's `\g@addto@macro`):

```
\makeatletter \g@addto@macro\UrlSpecials{\do\!{\newline}}
```

Here is another overly-complicated example to put extra flexible muglue around each "`/`" character, except when followed by another "`/`", as in "`http://`", where extra spacing looks poor.

```
% what we'll insert before and after each (lone) slash:
\newmuskip\Urlslashmuskip
\Urlslashmuskip=2mu plus2mu minus2mu

% change what / does:
\g@addto@macro\UrlSpecials{\do\/{\Urlspaceyslash}}

% need to look ahead:
\def\Urlspaceyslash{\futurelet\Urlssnext\finishUrlspaceyslash}

\def\finishUrlspaceyslash{%
  \mskip\Urlslashmuskip  % extra space before
  \mathchar8239  % "202f, i.e., binary op, \fam, / char
  % if we see //, eliminate the extra space to taste:
  \ifx\Urlssnext/\mskip-\Urlslashmuskip
  \else\mskip\Urlslashmuskip \fi
 }
```

If this sounds confusing ... well, it is! But I hope you won't need to redefine breakpoints — the default assignments seem to work well for a wide variety of applications. If you do need to make changes, you can test for breakpoints using regular math mode and the characters "`+=(a`".

6 Yet more flexibility

You can also customize the presentation of verbatim text by defining `\UrlRight` and/or `\UrlLeft`. An example for ISO formatting of urls surrounded by `<  >` is

```
\DeclareUrlCommand\url{\def\UrlLeft{<url:\ }\def\UrlRight{>}%
    \urlstyle{tt}}
```

The meanings of `\UrlLeft` and `\UrlRight` are *not* reproduced verbatim. This lets you use formatting commands there, but you must be careful not to use TEX's special characters (`\^_%~#$&{}` etc.) improperly. You can also define `\UrlLeft` to reprocess the verbatim text, but the format of the definition is special:

```
\def\UrlLeft#1\UrlRight{... do things with #1 ... }
```

Yes, that is `#1` followed by `\UrlRight` then the definition (a TEX macro with delimited arguments). For example, to produce a hyperTEX hypertext link:

```
\def\UrlLeft#1\UrlRight{%
      \special{html:<a href="#1">}#1\special{html:</a>}}
```

Using this technique, `url.sty` can provide a convenient interface for performing various operations on verbatim text. You don't even need to print out the argument! For greatest efficiency in such obscure applications, you can define a null url-style where all the lists like `\UrlBreaks` are empty.

Please note that this method is *not* how the hyperref package manages urls for its `\url` command, even though it makes use of `url.sty`. Instead, hyperref's `\url` reads its argument in a less-verbatim manner than described above, produces its hyperlink, and invokes `\nolinkurl` to format the text. `\nolinkurl` is the `\url` command descibed herein.