

The breakurl package*

Vilar Camara Neto
neto@dcc.ufmg.br

January 24, 2009

1 Introduction

The `hyperref` package brings a lot of interesting tools to “boost” documents produced by \LaTeX . For instance, a PDF file can have clickable links to references, section headings, URLs, etc.

Generating a link to a URL may be a concern if it stands near the end of a line of text. When one uses `pdf \LaTeX` to directly generate a PDF document, there’s no problem: the driver can break a link across more than one line. However, the `dvips` driver (used when one prefers the $\LaTeX \rightarrow \text{DVI} \rightarrow \text{PostScript} \rightarrow \text{PDF}$ path), because of internal reasons, can’t issue line breaks in the middle of a link. Sometimes this turns into a serious aesthetic problem, generating largely underfull/overfull paragraphs; sometimes the final effect is so bad that the link can’t even fit inside the physical page limits.

To overcome that `dvips` limitation, the `breakurl` package was designed with a simple solution: it provides a command called `\bur1` (it stands for “breakable URL”). Instead of generating one long, atomic link, this command breaks it into small pieces, allowing line breaks between them. Each sequence of pieces that stand together in one line are converted to a hot (clickable) link in the PDF document. Also, by default the `\url` command is turned into a synonym of `\bur1`, so there’s no need to do a search-replace operation to immediately start using the package.

2 How to use it

At the preamble, just put `\usepackage{breakurl}` somewhere *after* `\usepackage{hyperref}`. The `\bur1` command is defined and, by default, the package also turns the `\url` command into a synonym of `\bur1`. This might come in handy, for example, if you use `Bib \TeX` , your `.bib`-file has lots of `\url` commands and you don’t want to replace them by `\bur1`. If, for some reason, you want to preserve the original behavior of `\url` (i.e., it creates an unbreakable

*This document corresponds to `breakurl` v1.30, dated 2009/01/24.

link), you must supply the `preserveurlmacro` option to the package (see Section 2.1).

In the middle of the document, the syntax of `\burl` (and its synonym `\url`) is the same as the original `\url`: `\burl{⟨URL⟩}`, where `⟨URL⟩` is, of course, the address to point to. You don't need to care (escape) about special characters like `%`, `&`, `_`, and so on.

Another handy command is `\burlalt{⟨ActualURL⟩}{⟨DisplayedURL⟩}`, where `⟨ActualURL⟩` is the actual link and `⟨DisplayedURL⟩` is the link text to be displayed in the document. For consistency, `\urlalt` is a synonym of `\burlalt`, unless the `preserveurlmacro` package option is specified.¹

The default behavior of the package is to break the link after any sequence of these characters:

<code>'.'</code> (colon)	<code>'/'</code> (slash)	<code>'.'</code> (dot)
<code>'?'</code> (question mark)	<code>'#'</code> (hash)	<code>'&'</code> (ampersand)
<code>'_'</code> (underline)	<code>','</code> (comma)	<code>';'</code> (semicolon)
<code>'!'</code> (exclamation mark)		

and before occurrences of any of these:

`'%'` (percent sign)

Remember that (with exception of percent sign) breaks are only allowed *after* a sequence of these characters, so a link starting with `http://` will never break before the second slash.

Also note that I decided not to include the `'-'` (hyphen) character in the default lists. It's to avoid a possible confusion when someone encounters a break after a hyphen, e.g.:

Please visit the page at `http://internet-`
`page.com`, which shows...

Here comes the doubt: The author is pointing to `http://internet-page.com` or to `http://internetpage.com`? The `breakurl` package *never* adds a hyphen when a link is broken across lines — so, the first choice would be the right one —, but we can't assume that the reader knows this rule; so, I decided to disallow breaks after hyphens. Nevertheless, if you want to overcome my decision, use the `hyphenbreaks` option:

```
\usepackage[hyphenbreaks]{breakurl}
```

2.1 Package options

When using the `\usepackage` command, you can give some options to customize the package behavior. Possible options are explained below:

¹The `\burlalt` command resembles `\hyperref's \href`, but since it works in a different manner I decided not to call it "`\bhref`".

- **hyphenbreaks**
Instructs the package to allow line breaks after hyphens.
- **preserveurlmacro**
Instructs the package to leave the `\url` command exactly as it was before the package inclusion. Also, `\urlalt` isn't defined as a synonym of `\burlalt`. In either case (i.e., using `preserveurlmacro` or not), the breakable link is available via the `\burl` command.
- **vertfit=*(criterion)***
Establishes how the link rectangle's height (and depth) will behave against the corresponding URL text's vertical range. There are three options for *(criterion)*: `local` makes each rectangle fit tightly to its text's vertical range. This means that each line of a link broken across lines can have a rectangle with different vertical sizes. `global` first calculates the height (and depth) to enclose the entire link and preserves the measures, so the link maintains the vertical size across lines. `strut` goes even further and ensures that the rectangle's vertical range corresponds to `\strut`. With this option, rectangles in adjacent lines can overlap. The default is `vertfit=local`.

2.2 Additional comments

As stated in the introduction, the `breakurl` is designed for those compiling documents via \LaTeX , not pdf\LaTeX . In the latter case, the package doesn't (re)define the `\url` command: it only defines `\burl` to be a synonym of whatever `\url` is defined (e.g., via `url` or `hyperref` packages). Of course, `\burl` may behave differently compared to (non-pdf) \LaTeX , because then the system will use other rules to make line breaks, spacing, etc.

Also, this package was not designed to nor tested against other drivers: it's compatible with `dvips` only.

2.3 Changelog

(presented in reverse chronological order)

- v1.30** Breaks are now allowed before percent sign (%).
- v1.23** `\hypersetup` now works anywhere.
- v1.22** Corrected blank lines appearing inside tables.
- v1.21** `\burlalt` and the synonym `\urlalt` now work with `pdflatex`. Also, there are a couple of bug fixes (thank you again, Heiko).
- v1.20** An update was needed because `hyperref`'s internals were changed. (Thanks Heiko for sending the correction patch.) Troubleshooting now includes a note about `\sloppy`.

v1.10 A new command, `\burlalt` (and the synonym `\urlalt`), allows one to specify different values for actual and displayed link.

v1.01 Fixed a bug that was happening when a link is split across pages.

v1.00 The `\UrlLeft` and `\UrlRight` (defined and explained in the `url` package) are now partially supported. By “partially” I mean: although the original (`url.sty`’s) documentation allows defining `\UrlLeft` as a command with one argument (things such `\def\UrlLeft#1\UrlRight{do things with #1}`), this isn’t expected to work with `breakurl`. Please use only the basic definition, e.g.: `\def\UrlLeft{<url:\ } \def\UrlRight{>}`.

v0.04 Corrected a bug that prevented URLs to be in color, in despite of `hyperref`’s `colorlinks` and `urlcolor` options. Added an error message if `vertfit` parameter is invalid.

v0.03 The package was tested against `pdfTeX` engine (which may be the default for some `TeX` distributions). Introduced a new package option, `vertfit`.

v0.02 The main issue of the initial release — the odd-looking sequence of small links in the same line, if one uses `hyperref`’s link borders — was resolved: now the package generates only one rectangle per line. Also, breaks after hyphens, which weren’t allowed in the previous release, are now a users’ option. Finally, the package can be used with `pdfLATEX` (in this case, `\burl` is defined to be a synonym of the original `\url` command).

v0.01 Initial release.

2.4 Troubleshooting

Here comes a few notes about known issues:

- I received some comments saying that in some cases `breakurl` destroys the formatting of the document: the left/right margins aren’t respected, justification becomes weird, etc. In all these cases, the problems were corrected when other packages were upgraded, notably `xkeyval`.
- If your compilation issues the following error:

```
! Undefined control sequence.  
<argument> \headerps@out ...
```

then you need to specify the `dvips` driver as an option to the `hyperref` package, e.g.:

```
\usepackage[dvips]{hyperref}
```

However, this is related to old versions of `hyperref`: the package is able to automatically determine the driver in current versions. It’s probably better to update your `LATEX` system.


```

16 \expandafter\endinput
17 \fi

```

Since `breakurl` is an extension to `hyperref`, let's complain loudly if the latter was not yet loaded:

```

18 \@ifpackageloaded{hyperref}{}{%
19 \PackageError{breakurl}{The breakurl depends on hyperref package}%
20 {I can't do anything. Please type X <return>, edit the source file%
21 \MessageBreak
22 and add \string\usepackage\string{hyperref\string} before
23 \string\usepackage\string{breakurl\string}.}
24 \endinput
25 }

```

The package options are handled by `\newifs`, which are declared and initialised:

```

26 \newif\if@preserveurlmacro\@preserveurlmacrofalse
27 \newif\if@burl@fitstrut\@burl@fitstrutfalse
28 \newif\if@burl@fitglobal\@burl@fitglobalfalse

```

`\burl@toks` The `breakurl` package uses a token list to store characters and tokens until a break point is reached:

```

29 \newtoks\burl@toks

```

`\burl@charlistbefore`
`\burl@charlistafter`
`\burl@defifstructure` The following support routines are designed to build the conditional structure that is the kernel of `\burl`: comparing each incoming character with the list of “breakable” characters and taking decisions on that. This conditional structure is built by `\burl@defifstructure` — which is called only at the end of package loading, because the character list (stored in `\burl@charlistbefore`) can be modified by the `hyphenbreaks` option.

```

30 \let\burl@charlistbefore\empty
31 \let\burl@charlistafter\empty
32
33 \def\burl@addtocharlistbefore{\g@addto@macro\burl@charlistbefore}
34 \def\burl@addtocharlistafter{\g@addto@macro\burl@charlistafter}
35
36 \bgroup
37 \catcode'\&=12\relax
38 \hyper@normalise\burl@addtocharlistbefore{%}
39 \hyper@normalise\burl@addtocharlistafter{:/.#&_ ;!}
40 \egroup
41
42 \def\burl@growmif#1#2{%
43 \g@addto@macro\burl@mif{\def\burl@ttt{#1}\ifx\burl@ttt\@nextchar#2\else}%
44 }
45 \def\burl@growmfi{%
46 \g@addto@macro\burl@mfi{\fi}%
47 }
48 \def\burl@defifstructure{%
49 \let\burl@mif\empty

```

```

50 \let\burl@mfi\empty
51 \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
52   \burl@charlistbefore\do{%
53     \expandafter\burl@growmif\@nextchar\@burl@breakbeforetrue
54     \burl@growmfi
55   }%
56 \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
57   \burl@charlistafter\do{%
58     \expandafter\burl@growmif\@nextchar\@burl@breakaftertrue
59     \burl@growmfi
60   }%
61 }
62
63 \AtEndOfPackage{\burl@defifstructure}

```

The package options are declared and handled as follows:

```

64 \def\burl@setvertfit#1{%
65   \lowercase{\def\burl@temp{#1}}%
66   \def\burl@opt{local}\ifx\burl@temp\burl@opt
67     \@burl@fitstrutfalse\@burl@fitglobalfalse
68   \else\def\burl@opt{strut}\ifx\burl@temp\burl@opt
69     \@burl@fitstruttrue\@burl@fitglobalfalse
70   \else\def\burl@opt{global}\ifx\burl@temp\burl@opt
71     \@burl@fitstrutfalse\@burl@fitglobaltrue
72   \else
73     \PackageWarning{breakurl}{Unrecognized vertfit option ‘\burl@temp’.%}
74     \MessageBreak
75     Adopting default ‘local’}
76   \@burl@fitstrutfalse\@burl@fitglobalfalse
77   \fi\fi\fi
78 }
79
80 \DeclareOptionX{preserveurlmacro}{\@preserveurlmacrotrue}
81 \DeclareOptionX{hyphenbreaks}{%
82   \bgroup
83     \catcode‘\&=12\relax
84     \hyper@normalise\burl@addtocharlistafter{-}%
85   \egroup
86 }
87 \DeclareOptionX{vertfit}[local]{\burl@setvertfit{#1}}
88
89 \ProcessOptionsX\relax

```

These supporting routines are modified versions of those found in the hyperref package. They were adapted to allow a link to be progressively built, i.e., when we say “put a link rectangle here”, the package will decide if this will be made.

```

90 \def\burl@hyper@linkurl#1#2{%
91   \begingroup
92     \hyper@chars
93     \burl@condpdflink{#1}%

```

```

94 \endgroup
95 }
96
97 \def\burl@condpdflink#1{%
98 \literalps@out{
99 /burl@bordercolor {\@urlbordercolor} def
100 /burl@border {\@pdfborder} def
101 }%
102 \if@burl@fitstrut
103 \sbox\pdf@box{#1\strut}%
104 \else\if@burl@fitglobal
105 \sbox\pdf@box{\burl@url}%
106 \else
107 \sbox\pdf@box{#1}%
108 \fi\fi
109 \dimen@ \ht\pdf@box\dimen@ii\dp\pdf@box
110 \sbox\pdf@box{#1}%
111 \ifdim\dimen@ii=\z@
112 \literalps@out{BU.SS}%
113 \else
114 \lower\dimen@ii\hbox{\literalps@out{BU.SS}}%
115 \fi
116 \ifHy@breaklinks\unhbox\else\box\fi\pdf@box
117 \ifdim\dimen@=\z@
118 \literalps@out{BU.SE}%
119 \else
120 \raise\dimen@\hbox{\literalps@out{BU.SE}}%
121 \fi
122 \pdf@addtoksx{H.B}%
123 }

```

`\burl` `\burl` prepares the catcodes (via `\hyper@normalise`) and calls the `\burl@` macro, which does the actual work.

```

124 \DeclareRobustCommand*\burl}{%
125 \leavevmode
126 \begingroup
127 \let\hyper@linkurl=\burl@hyper@linkurl
128 \catcode'\&=12\relax
129 \hyper@normalise\burl@
130 }

```

`\burlalt` `\burlalt` does the same as `\burl`, but calls another macro (`\burl@alt`) to read two following arguments instead of only one.

```

131 \DeclareRobustCommand*\burlalt}{%
132 \begingroup
133 \let\hyper@linkurl=\burl@hyper@linkurl
134 \catcode'\&=12\relax
135 \hyper@normalise\burl@alt
136 }

```


`\burl@` `\burl@` $\langle URL \rangle$ just eats the next argument to define the URL address and
`\burl@alt` the link to be displayed. Both are used by `\burl@doit`.
`\burl@@alt` `\burl@alt` $\langle ActualURL \rangle$ and `\burl@@alt` $\langle DisplayedURL \rangle$ work together
to eat the two arguments (the actual URL to point to and the link text to be
displayed). Again, both are used by `\burl@doit`.

```

137 \newif\if@burl@breakbefore
138 \newif\if@burl@breakafter
139 \newif\if@burl@prevbreakafter
140
141 \bgroup
142 \catcode'\&=12\relax
143 \gdef\burl@#1{%
144   \def\burl@url{#1}%
145   \def\burl@urltext{#1}%
146   \burl@doit
147 }
148
149 \gdef\burl@alt#1{%
150   \def\burl@url{#1}%
151   \hyper@normalise\burl@@alt
152 }
153 \gdef\burl@@alt#1{%
154   \def\burl@urltext{#1}%
155   \burl@doit
156 }

```

`\burl@doit` `\burl@doit` works much like `hyperref`'s `\url@` macro (actually, this code macro
was borrowed and adapted from the original `\url@`): it builds a series of links,
allowing line breaks between them. The characters are accumulated and eventually
flushed via the `\burl@flush` macro.

Support for `\UrlLeft`/`\UrlRight`: The `\UrlRight` is emptied until the very
last flush (when it is restored). The `\UrlLeft` is emptied after the first flush.
So, any string defined in those macros are meant to be displayed only before the
first piece and after the last one, which (of course) is what we expect to happen.
Unfortunately, breaking doesn't happen inside those strings, since they're not
rendered verbatim (and so they aren't processed inside the breaking mechanism).

```

157 \gdef\burl@doit{%
158   \burl@toks{}%
159   \let\burl@UrlRight\UrlRight
160   \let\UrlRight\empty
161   \@burl@prevbreakafterfalse
162   \@ifundefined{@urlcolor}{\Hy@colorlink\@linkcolor}{\Hy@colorlink\@urlcolor}%
163   \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
164   \burl@urltext\do{%
165     \if@burl@breakafter\@burl@prevbreakaftertrue
166     \else\@burl@prevbreakafterfalse\fi
167   \@burl@breakbeforefalse
168   \@burl@breakafterfalse

```

```

169 \expandafter\burl@mif\burl@mfi
170 \if@burl@breakbefore
171 % Breakable if the current char is in the ‘can break before’ list
172 \burl@flush\linebreak[0]%
173 \else
174 \if@burl@prevbreakafter
175 \if@burl@breakafter\else
176 % Breakable if the current char is not in any of the ‘can break’
177 % lists, but the previous is in the ‘can break after’ list.
178 % This mechanism accounts for sequences of ‘break after’ characters,
179 % where a break is allowed only after the last one
180 \burl@flush\linebreak[0]%
181 \fi
182 \fi
183 \fi
184 \expandafter\expandafter\expandafter\burl@toks
185 \expandafter\expandafter\expandafter{%
186 \expandafter\the\expandafter\burl@toks\@nextchar}%
187 }%
188 \let\UrlRight\burl@UrlRight
189 \burl@flush
190 \literalps@out{BU.E}%
191 \Hy@endcolorlink
192 \endgroup
193 }
194 \egroup

```

`\burl@flush` This macro flushes the characters accumulated during the `\burl@` processing, creating a link to the URL.

```

195 \def\the@burl@toks{\the\burl@toks}
196
197 \def\burl@flush{%
198 \expandafter\def\expandafter\burl@toks@def\expandafter{\the\burl@toks}%
199 \literalps@out{/BU.L (\burl@url) def}%
200 \hyper@linkurl{\expandafter\Hurl\expandafter{\burl@toks@def}}{\burl@url}%
201 \global\burl@toks{}}%
202 \let\UrlLeft\empty
203 }%

```

Now the synonyms `\url` and `\urlalt` are (re)defined, unless the `preserveurlmacro` option is given.

```

204 \if@preserveurlmacro\else\let\url\burl\let\urlalt\burlalt\fi

```

Internally, the package works as follows: each link segment (i.e., a list of non-breakable characters followed by breakable characters) ends with a PDF command that checks if the line ends here. If this check is true, then (and only then) the PDF link rectangle is built, embracing all link segments of this line.

To make that work, we need some code to work at the PostScript processing level. The supporting routines to do so are introduced in the PS dictionary initialization block via specials. Each routine is explained below.

The variables used here are: `burl@stx` and `burl@endx`, which defines the link's horizontal range; `burl@boty` and `burl@topy`, which defines the link's vertical range; `burl@llx`, `burl@lly`, `burl@urx`, and `burl@ury`, which define the bounding box of the current link segment (they resemble the `hyperref`'s `pdf@llx`–`pdf@ury` counterparts); and `BU.L`, which holds the target URL.

```
205 \AtBeginDvi{%
206   \headerps@out{%
207     /burl@stx null def
```

`BU.S` is called whenever a link begins:

```
208   /BU.S {
209     /burl@stx null def
210   } def
```

`BU.SS` is called whenever a link segment begins:

```
211   /BU.SS {
212     currentpoint
213     /burl@lly exch def
214     /burl@llx exch def
215     burl@stx null ne {burl@endx burl@llx ne {BU.FL BU.S} if} if
216     burl@stx null eq {
217       burl@llx dup /burl@stx exch def /burl@endx exch def
218       burl@lly dup /burl@boty exch def /burl@topy exch def
219     } if
220     burl@lly burl@boty gt {/burl@boty burl@lly def} if
221   } def
```

`BU.SE` is called whenever a link segment ends:

```
222   /BU.SE {
223     currentpoint
224     /burl@ury exch def
225     dup /burl@urx exch def /burl@endx exch def
226     burl@ury burl@topy lt {/burl@topy burl@ury def} if
227   } def
```

`BU.SE` is called whenever the entire link ends:

```
228   /BU.E {
229     BU.FL
230   } def
```

`BU.FL` is called to conditionally flush the group of link segments that we have so far. This is meant to be called at each line break:

```
231   /BU.FL {
232     burl@stx null ne {BU.DF} if
233   } def
```

`BU.DF` is the routine to actually put the link rectangle in the PDF file:

```
234   /BU.DF {
235     BU.BB
```

```

236     [ /H /I /Border [burl@border] /Color [burl@bordercolor]
237     /Action << /Subtype /URI /URI BU.L >> /Subtype /Link BU.B /ANN pdfmark
238     /burl@stx null def
239   } def

```

BU.FF adds margins to the calculated tight rectangle:

```

240   /BU.BB {
241     burl@stx HyperBorder sub /burl@stx exch def
242     burl@endx HyperBorder add /burl@endx exch def
243     burl@boty HyperBorder add /burl@boty exch def
244     burl@topy HyperBorder sub /burl@topy exch def
245   } def

```

BU.B converts the coordinates into a rectangle:

```

246   /BU.B {
247     /Rect[burl@stx burl@boty burl@endx burl@topy]
248   } def

```

Finally, we must redefine `eop`, which is called just when the page ends, to handle links that are split across pages. (`eop-hook` isn't the right place to do so, since this hook is called after the dictionaries were reverted to a previous state, vanishing the rectangle coordinates.)

```

249   /eop where {
250     begin
251     /@ldeopburl /eop load def
252     /eop { SDict begin BU.FL end @ldeopburl } def
253     end
254   } {
255     /eop { SDict begin BU.FL end } def
256   } ifelse
257 }%
258 }

```