

The breakurl package*

Vilar Camara Neto
neto@dcc.ufmg.br

August 26, 2006

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 the document is being generated by $\text{pdf}\text{\LaTeX}$, there’s no problem: the driver can break a link across more than one line. However, the `dvips` driver (used when one prefers the $\text{\LaTeX} \rightarrow \text{DVI} \rightarrow \text{PostScript} \rightarrow \text{PDF}$ 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 `\burl` (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.

2 How to use it

At the preamble, just put a `\usepackage{breakurl}` somewhere *after* the `\usepackage{hyperref}`. The `\burl` command is defined and, by default, the package also turns the `\url` command into a synonym of `\burl`. This might come in handy, for example, if you use $\text{Bib}\text{\TeX}$, your `.bib`-file has lots of `\url` commands and you don’t want to replace them by `\burl`. If, for some reason, you want to preserve the original behavior of `\url` (i.e., it creates an unbreakable link), you must supply the `preserveurlmacro` option to the package (see Section 2.1).

*This document corresponds to `breakurl` v1.20, dated 2006/08/26.

In the middle of the document, the syntax of `\burl` (and its synonym `\url`) is exactly like 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:

‘:’ (colon)	‘/’ (slash)	‘.’ (dot)
‘?’ (question mark)	‘#’ (hash)	‘&’ (ampersand)
‘_’ (underline)	‘,’ (comma)	‘;’ (semicolon)
‘!’ (exclamation mark)		

You can add ‘-’ (hyphen) to this list (see below), but read why I decided to keep it out of the default list.

Remember that breaks are 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 this default list. 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:

- **hyphenbreaks**

Instructs the package to allow line breaks after hyphens.

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

- **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.

- **verffit=*< 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 `verffit=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. And don't forget to specify the driver (`dvips`) to the `hyperref` package, i.e.:

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

2.3 Changelog

(presented in reverse chronological order)

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 into more than one page.

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 `verffit` parameter is invalid.
- v0.03** The package was tested against `pdfeTeX` engine (which may be the default for some `teTeX` distributions). Introduced a new package option, `verffit`.
- 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 ...
```

it's probably because you forgot to specify the `dvips` driver as an option to `hyperref`. Check if you have it:

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

- If everything compiles but sometimes URLs still don't respect the right margin, don't blame the package yet :-). Roughly speaking, by default the right margin is a limit to be respected "only if word spacing is okay", so it may be ignored even when URLs aren't used. Check the following paragraph:

[illegible]

To overcome this (and make right margins a hard limit) use the command `\sloppy`, preferably before `\begin{document}`. This makes the previous paragraph look like:


```

15 \expandafter\gdef\expandafter\burl@charlist\expandafter{%
16   \burl@charlist #1}%
17 }
18
19 \bgroup
20 \catcode'\&=12\relax
21 \hyper@normalise\burl@addtocharlist{./?#&_;!}
22 \egroup
23
24 \def\burl@growmif#1{%
25   \expandafter\def\expandafter\burl@mif\expandafter{%
26     \burl@mif\def\burl@ttt{#1}\ifx\burl@ttt\@nextchar\@burl@breakabletrue\else
27   }%
28 }
29 \def\burl@growmfi{%
30   \expandafter\def\expandafter\burl@mfi\expandafter{\burl@mfi\fi}%
31 }
32 \def\burl@melse{%
33   \if@burl@breakable\burl@flush\linebreak[0]\@burl@breakablefalse\fi
34   \expandafter\expandafter\expandafter\burl@toks
35     \expandafter\expandafter\expandafter{%
36     \expandafter\the\expandafter\burl@toks\@nextchar}%
37 }
38 \def\burl@defifstructure{%
39   \def\burl@mif{}%
40   \def\burl@mfi{}%
41   \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
42     \burl@charlist\do{%
43     \expandafter\burl@growmif\@nextchar
44     \burl@growmfi
45   }%
46 }
47
48 \AtEndOfPackage{\burl@defifstructure}

```

The package options are declared and handled as follows:

```

49 \def\burl@setverffit#1{%
50   \lowercase{\def\burl@temp{#1}}%
51   \def\burl@opt{local}\ifx\burl@temp\burl@opt
52     \@burl@fitstrutfalse\@burl@fitglobalfalse
53   \else\def\burl@opt{strut}\ifx\burl@temp\burl@opt
54     \@burl@fitstruttrue\@burl@fitglobalfalse
55   \else\def\burl@opt{global}\ifx\burl@temp\burl@opt
56     \@burl@fitstrutfalse\@burl@fitglobaltrue
57   \else
58     \PackageWarning{breakurl}{Unrecognized verffit option '\burl@temp'.^^J%
59     Adopting default 'local'}
60     \@burl@fitstrutfalse\@burl@fitglobalfalse
61   \fi\fi\fi
62 }

```

```

63
64 \DeclareOptionX{preserveurlmacro}{\@preserveurlmacrotrue}
65 \DeclareOptionX{hyphenbreaks}{\bgroup
66   \catcode'\&=12\relax\hyper@normalise\burl@addtocharlist{-}%
67   \egroup}
68 \DeclareOptionX{vertfit}[local]{\burl@setvertfit{#1}}
69
70 \ProcessOptionsX\relax

```

Is the document being processed by pdf^LA_TE_X? Then, well, this package doesn't apply: let's just define `\burl` to call the default `\url`.

```

71 \ifx\pdfoutput\undefined\else\ifx\pdfoutput\relax\else\ifcase\pdfoutput\else
72   \PackageWarning{breakurl}{%
73     You are using breakurl while processing via pdfLatEX.^^J%
74     \string\burl\space will be just a synonym of \string\url.^^J}
75   \DeclareRobustCommand{\burl}{\url}
76   \endinput
77 \fi\fi\fi

```

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.

```

78 \def\burl@hyper@linkurl#1#2{%
79   \begingroup
80     \hyper@chars
81     \leavevmode
82     \burl@condpdf@link{#1}%
83   \endgroup
84 }
85
86 \def\burl@condpdf@link#1{%
87   \leavevmode
88   \if@burl@fit@strut
89     \sbox\pdf@box{#1\strut}%
90   \else\if@burl@fit@global
91     \sbox\pdf@box{\burl@url}%
92   \else
93     \sbox\pdf@box{#1}%
94   \fi\fi
95   \dimen@ht\pdf@box\dimen@ii\dp\pdf@box
96   \sbox\pdf@box{#1}%
97   \leavevmode
98   \ifdim\dimen@ii=\z@
99     \literalps@out{BU.SS}%
100  \else
101    \lower\dimen@ii\hbox{\literalps@out{BU.SS}}%
102  \fi
103  \ifHy@breaklinks\unhbox\else\box\fi\pdf@box
104  \ifdim\dimen@=\z@
105    \literalps@out{BU.SE}%

```

```

106 \else
107   \raise\dimen@\hbox{\literalps@out{BU.SE}}}%
108 \fi
109 \pdf@addtoksx{H.B}%
110 }

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

111 \DeclareRobustCommand*\ Burl@{%
112   \begingroup
113   \let\hyper@linkurl=\ Burl@hyper@linkurl
114   \catcode'\&=12\relax
115   \hyper@normalise\ Burl@
116 }

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

117 \DeclareRobustCommand*\ Burlalt{%
118   \begingroup
119   \let\hyper@linkurl=\ Burl@hyper@linkurl
120   \catcode'\&=12\relax
121   \hyper@normalise\ Burl@alt
122 }

\ Burl@ \ Burl@ {<URL>} 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 {<ActualURL>} and \ Burl@@alt {<DisplayedURL>} 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.

123 \newif\if@ Burl@breakable
124
125 \bgroup
126 \catcode'\&=12\relax
127 \gdef\ Burl@#1{%
128   \def\ Burl@url{#1}%
129   \def\ Burl@urltext{#1}%
130   \ Burl@doit
131 }
132
133 \gdef\ Burl@alt#1{%
134   \def\ Burl@url{#1}%
135   \hyper@normalise\ Burl@alt
136 }
137 \gdef\ Burl@@alt#1{%
138   \def\ Burl@urltext{#1}%
139   \ Burl@doit
140 }

\ 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).

```

141 \gdef\burl@doit{%
142   \burl@toks{}%
143   \let\burl@UrlRight=\UrlRight
144   \let\UrlRight=\empty
145   \ifundefined{@urlcolor}{\Hy@colorlink\@linkcolor}{\Hy@colorlink\@urlcolor}%
146   \@burl@breakablefalse
147   \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
148     \burl@urltext\do{%
149     \expandafter\burl@mif\expandafter\burl@melse\burl@mfi
150     \if@burl@breakable
151       \expandafter\expandafter\expandafter\burl@toks
152       \expandafter\expandafter\expandafter{%
153       \expandafter\the\expandafter\burl@toks\@nextchar}%
154     \fi
155   }%
156   \let\UrlRight=\burl@UrlRight
157   \burl@flush
158   \literalps@out{BU.E}%
159   \Hy@endcolorlink
160   \endgroup
161 }
162 \egroup

```

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

```

163 \def\the@burl@toks{\the\burl@toks}
164
165 \def\burl@flush{%
166   \expandafter\def\expandafter\burl@toks\def\expandafter{\the\burl@toks}%
167   \literalps@out{/BU.L (\burl@url) def}%
168   \hyper@linkurl{\expandafter\Hurl\expandafter{\burl@toks@def}}{\burl@url}%
169   \global\burl@toks{}%
170   \let\UrlLeft=\empty
171 }%

```

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

```

172 \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 PDF processing level. The supporting routines to do so are introduced in the PDF 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.

```
173 \AtBeginDvi{%
174   \headerps@out{%
175     /burl@stx null def
```

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

```
176   /BU.S {
177     /burl@stx null def
178   } def
```

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

```
179   /BU.SS {
180     currentpoint
181     /burl@lly exch def
182     /burl@llx exch def
183     burl@stx null ne {burl@endx burl@llx ne {BU.FL BU.S} if} if
184     burl@stx null eq {
185       burl@llx dup /burl@stx exch def /burl@endx exch def
186       burl@lly dup /burl@boty exch def /burl@topy exch def
187     } if
188     burl@lly burl@boty gt {/burl@boty burl@lly def} if
189   } def
```

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

```
190   /BU.SE {
191     currentpoint
192     /burl@ury exch def
193     dup /burl@urx exch def /burl@endx exch def
194     burl@ury burl@topy lt {/burl@topy burl@ury def} if
195   } def
```

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

```
196   /BU.E {
197     BU.FL
198   } 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:

```
199   /BU.FL {
200     burl@stx null ne {BU.DF} if
201   } def
```

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

```
202   /BU.DF {
203     BU.BB
204     [ /H /I /Border [\@pdfborder] /Color [\@urlbordercolor]
205     /Action << /Subtype /URI /URI BU.L >> /Subtype /Link BU.B /ANN pdfmark
206     /burl@stx null def
207   } def
```

BU.FF adds margins to the calculated tight rectangle:

```
208   /BU.BB {
209     burl@stx HyperBorder sub /burl@stx exch def
210     burl@endx HyperBorder add /burl@endx exch def
211     burl@boty HyperBorder add /burl@boty exch def
212     burl@topy HyperBorder sub /burl@topy exch def
213   } def
```

BU.B converts the coordinates into a rectangle:

```
214   /BU.B {
215     /Rect[burl@stx burl@boty burl@endx burl@topy]
216   } def
```

Finally, we must redefine `eop`, which is called just when the page ends, to handle links that are split into more than one page. (`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.)

```
217   /eop where {
218     begin
219     /\ldeopburl /eop load def
220     /eop { SDict begin BU.FL end @ldeopburl } def
221     end
222   } {
223     /eop { SDict begin BU.FL end } def
224   } ifelse
225 }%
226 }
```