

The breakurl package*

Vilar Camara Neto
neto@dcc.ufmg.br

December 18, 2007

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 $\text{pdf}\text{\LaTeX}$ 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 $\text{\LaTeX} \rightarrow \text{DVI} \rightarrow \text{PostScript} \rightarrow \text{PDF}$ 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` v1.21, dated 2007/12/18.

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)		

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.

- **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. 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)

v1.21 `\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 into more than one page.

[illegible]

Acknowledgments

3 Source code

The `breakurl` requires some packages, so let's include them:

```

3 \ifpdf
4   % Dummy package options
5   \DeclareOptionX{preserveurlmacro}{}
6   \DeclareOptionX{hyphenbreaks}{}
7   \DeclareOptionX{verffit}{}
8   \ProcessOptionsX\relax
9
10  \PackageWarning{breakurl}{%
11    You are using breakurl while processing via pdflatex.\MessageBreak
12    \string\burl\space will be just a synonym of \string\url.\MessageBreak}
13  \DeclareRobustCommand{\burl}{\url}
14  \DeclareRobustCommand*{\burlalt}{\hyper@normalise\burl@alt}
15  \def\burl@alt#1#2{\hyper@linkurl{\Hurl{#1}}{#2}}
16  \expandafter\endinput
17 \fi

```

```
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@charlist` The following support routines are designed to build the conditional structure
`\burl@defifstructure` 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@charlist`) can be modified by the `hyphenbreaks` option.

```

30 \def\burl@charlist{}
31 \def\burl@addtocharlist#1{%
32   \expandafter\gdef\expandafter\burl@charlist\expandafter{%
33     \burl@charlist #1}%
34 }
35
36 \bgroup
37 \catcode'\&=12\relax
38 \hyper@normalise\burl@addtocharlist{:/.?#&_;;!}
39 \egroup
40
41 \def\burl@growmif#1{%
42   \expandafter\def\expandafter\burl@mif\expandafter{%
43     \burl@mif\def\burl@ttt{#1}\ifx\burl@ttt\@nextchar\@burl@breakabletrue\else
44   }%
45 }
46 \def\burl@growmfi{%
47   \expandafter\def\expandafter\burl@mfi\expandafter{\burl@mfi\fi}%
48 }
49 \def\burl@melse{%
50   \if@burl@breakable\burl@flush\linebreak[0]\@burl@breakablefalse\fi
51   \expandafter\expandafter\expandafter\burl@toks
52     \expandafter\expandafter\expandafter{%
53     \expandafter\the\expandafter\burl@toks\@nextchar}%
54 }
55 \def\burl@defifstructure{%

```

```

56 \def\burl@mif{}%
57 \def\burl@mfi{}%
58 \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
59 \burl@charlist\do{%
60 \expandafter\burl@growmif\@nextchar
61 \burl@growmfi
62 }%
63 }
64
65 \AtEndOfPackage{\burl@defifstructure}

```

The package options are declared and handled as follows:

```

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

```

89 \def\burl@hyper@linkurl#1#2{%
90 \begingroup
91 \hyper@chars
92 \leavevmode
93 \burl@condpdflink{#1}%
94 \endgroup
95 }
96
97 \def\burl@condpdflink#1{%
98 \leavevmode
99 \if@burl@fitstrut

```

```

100     \sbox\pdf@box{#1\strut}%
101 \else\if@burl@fitglobal
102     \sbox\pdf@box{\burl@url}%
103 \else
104     \sbox\pdf@box{#1}%
105 \fi\fi
106 \dimen@ \ht\pdf@box\dimen@ii\dp\pdf@box
107 \sbox\pdf@box{#1}%
108 \leavevmode
109 \ifdim\dimen@ii=\z@
110     \literalps@out{BU.SS}%
111 \else
112     \lower\dimen@ii\hbox{\literalps@out{BU.SS}}%
113 \fi
114 \ifHy@breaklinks\unhbox\else\box\fi\pdf@box
115 \ifdim\dimen@=\z@
116     \literalps@out{BU.SE}%
117 \else
118     \raise\dimen@\hbox{\literalps@out{BU.SE}}%
119 \fi
120 \pdf@addtoksx{H.B}%
121 }

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

122 \DeclareRobustCommand*\burl{%
123     \begingroup
124     \let\hyper@linkurl=\burl@hyper@linkurl
125     \catcode'\&=12\relax
126     \hyper@normalise\burl@
127 }

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

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

\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.

134 \newif\if@burl@breakable
135
136 \bgroup

```



```

137 \catcode'\&=12\relax
138 \gdef\burl@#1{%
139   \def\burl@url{#1}%
140   \def\burl@urltext{#1}%
141   \burl@doit
142 }
143
144 \gdef\burl@alt#1{%
145   \def\burl@url{#1}%
146   \hyper@normalise\burl@@alt
147 }
148 \gdef\burl@@alt#1{%
149   \def\burl@urltext{#1}%
150   \burl@doit
151 }

```

`\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).

```

152 \gdef\burl@doit{%
153   \burl@toks{}%
154   \let\burl@UrlRight=\UrlRight
155   \let\UrlRight=\empty
156   \ifundefined{@urlcolor}{\Hy@colorlink\@linkcolor}{\Hy@colorlink\@urlcolor}%
157   \@burl@breakablefalse
158   \expandafter\@tfor\expandafter\@nextchar\expandafter:\expandafter=%
159     \burl@urltext\do{%
160       \expandafter\burl@mif\expandafter\burl@melse\burl@mfi
161       \if@burl@breakable
162         \expandafter\expandafter\expandafter\burl@toks
163         \expandafter\expandafter\expandafter{%
164           \expandafter\the\expandafter\burl@toks\@nextchar}%
165       \fi
166     }%
167   \let\UrlRight=\burl@UrlRight
168   \burl@flush
169   \literalps@out{BU.E}%
170   \Hy@endcolorlink
171   \endgroup
172 }
173 \egroup

```

`\burl@flush` This macro flushes the characters accumulated during the `\burl@` processing,

creating a link to the URL.

```

174 \def\the@burl@toks{\the\burl@toks}
175
176 \def\burl@flush{%
177   \expandafter\def\expandafter\burl@toks@def\expandafter{\the\burl@toks}%
178   \literalps@out{/BU.L (\burl@url) def}%
179   \hyper@linkurl{\expandafter\Hurl\expandafter{\burl@toks@def}}{\burl@url}%
180   \global\burl@toks{}}%
181   \let\UrlLeft=\empty
182 }%

```

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

```

183 \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.

```

184 \AtBeginDvi{%
185   \headerps@out{%
186     /burl@stx null def

```

BU.S is called whenever a link begins:

```

187   /BU.S {
188     /burl@stx null def
189   } def

```

BU.SS is called whenever a link segment begins:

```

190   /BU.SS {
191     currentpoint
192     /burl@lly exch def
193     /burl@llx exch def
194     burl@stx null ne {burl@endx burl@llx ne {BU.FL BU.S} if} if
195     burl@stx null eq {
196       burl@llx dup /burl@stx exch def /burl@endx exch def
197       burl@lly dup /burl@boty exch def /burl@topy exch def
198     } if
199     burl@lly burl@boty gt {/burl@boty burl@lly def} if
200   } def

```

BU.SE is called whenever a link segment ends:

```
201 /BU.SE {
202   currentpoint
203   /burl@ury exch def
204   dup /burl@urx exch def /burl@endx exch def
205   burl@ury burl@topy lt {/burl@topy burl@ury def} if
206 } def
```

BU.SE is called whenever the entire link ends:

```
207 /BU.E {
208   BU.FL
209 } 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:

```
210 /BU.FL {
211   burl@stx null ne {BU.DF} if
212 } def
```

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

```
213 /BU.DF {
214   BU.BB
215   [ /H /I /Border [ \@pdfborder ] /Color [ \@urlbordercolor ]
216   /Action << /Subtype /URI /URI BU.L >> /Subtype /Link BU.B /ANN pdfmark
217   /burl@stx null def
218 } def
```

BU.FF adds margins to the calculated tight rectangle:

```
219 /BU.BB {
220   burl@stx HyperBorder sub /burl@stx exch def
221   burl@endx HyperBorder add /burl@endx exch def
222   burl@boty HyperBorder add /burl@boty exch def
223   burl@topy HyperBorder sub /burl@topy exch def
224 } def
```

BU.B converts the coordinates into a rectangle:

```
225 /BU.B {
226   /Rect[burl@stx burl@boty burl@endx burl@topy]
227 } 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.)

```
228 /eop where {
229   begin
230   /@ldeopburl /eop load def
231   /eop { SDict begin BU.FL end @ldeopburl } def
232   end
```

```
233    } {  
234        /eop { SDict begin BU.FL end } def  
235    } ifelse  
236 }%  
237 }
```