The wheretotrim package*

Scott Pakin
scott+wtt@pakin.org

May 15, 2013

1 Introduction

`wheretotrim` is a tool to help LaTeX users reduce their document's page count. It is intended to be used with documents that exceed a publisher's specified page-length limitation by a small amount (much less than a full column or page). `wheretotrim` is therefore similar to the savetrees package in that both seek to reduce page count. The two differ in that savetrees saves space by altering document formatting while `wheretotrim` suggests where text can be removed to reduce page count without altering any formatting. `wheretotrim` and savetrees are compatible with each other, though: `wheretotrim` can safely be run on a document that includes a `\usepackage{savetrees}`.

`wheretotrim` operates by building the document repeatedly, successively expanding each column on each page by one line height to mimic reducing the amount of text in that column by an equivalent amount. If doing so does not reduce the page count, `wheretotrim` repeats the process with two line heights' expansion of each column, then three, and so forth until it expands each column in turn by the full height of the column. The following is some sample output for a single-column document when `wheretotrim` is run with the **--allpages** option (cf. Section 2.1):

```
To reduce the page count from 11 to 10, do any of the following:

  * Reduce page 2 by 8 lines.
  * Reduce page 5 by 7 lines.
  * Reduce page 6 by 7 lines.
  * Reduce page 7 by 7 lines.
  * Reduce page 8 by 7 lines.
  * Reduce page 9 by 7 lines.
  * Reduce page 10 by 5 lines.
  * Reduce page 11 by 5 lines.

Note: 5 lines = 1.0" = 2.5 cm = 11.1% of the page height
```

*This document corresponds to wheretotrim v1.0, dated 2013/05/15.

That is, reducing either page 10 or page 11 by five lines is the most expedient way to reduce the document's page count. Seven lines would need to be cut from page 5, 6, 7, or 8 to achieve the same effect, and eight lines would need to be cut from page 2. In contrast, no amount of text trimming on pages 1, 3, or 4 will reduce the page count.

2 Usage

Sections 2.1–2.4 explain how to use `wheretotrim`.

2.1 Command-line options

Run `wheretotrim` as follows:

> `wheretotrim` [**--allpages**] [**--verbose** | **--quiet**]
> [**--debug**=⟨*page*⟩,⟨*column*⟩,⟨*lines*⟩] ⟨*latex command*⟩

or

> `wheretotrim` [**--verbose**] **--help** | **--version**

`wheretotrim` accepts the following command-line options:

-a, --allpages Perform enough extra runs of `latex` to report the amount of space that must be trimmed from *each* column or page to reduce page count, not just the columns or pages to which the page count is the most sensitive.

-v, --verbose Display the output of each run of `latex`. This is useful for troubleshooting and to help monitor the progress of long `latex` runs.

-q, --quiet Suppress progress updates and output only the final report.

-d ⟨*page*⟩,⟨*column*⟩,⟨*lines*⟩, --debug=⟨*page*⟩,⟨*column*⟩,⟨*lines*⟩ Debug `wheretotrim`'s execution by expanding page ⟨*page*⟩, column ⟨*column*⟩ by ⟨*lines*⟩ line heights and leaving the `latex` output in that state.

-h, --help Summarize usage information and exit. These may be used with **--verbose** to display more extended documentation.

-V, --version Display `wheretotrim`'s version number and exit.

In addition to the preceding options, `wheretotrim` requires a ⟨*latex command*⟩ argument that specifies how to build the document.

2.2 Examples

For the most basic usage, simply provide a `latex` command to run:

```
wheretotrim latex myfile.tex
```

or, for example,

```
wheretotrim pdflatex myfile.tex
```

`wheretotrim` executes the specified command a large number of times and finally terminates with a report resembling the following:

```
To reduce the page count from 10 to 9, do any of the following:

  * Reduce page 9, column 1 by 12 lines.
  * Reduce page 9, column 2 by 12 lines.
  * Reduce page 10, column 1 by 12 lines.

Note: 12 lines = 2.4" = 6.1 cm = 26.8% of the column height
```

To ask `wheretotrim` to report how much space needs to be trimmed on each column and page to reduce the total page count, specify the **--allpages** option:

```
wheretotrim --allpages latex myfile.tex
```

The output now looks like the following:

```
To reduce the page count from 10 to 9, do any of the following:

  * Reduce page 1, column 1 by 13 lines.
  * Reduce page 1, column 2 by 13 lines.
  * Reduce page 2, column 1 by 13 lines.
  * Reduce page 2, column 2 by 13 lines.
  * Reduce page 4, column 1 by 13 lines.
  * Reduce page 4, column 2 by 13 lines.
  * Reduce page 5, column 1 by 13 lines.
  * Reduce page 5, column 2 by 13 lines.
  * Reduce page 6, column 1 by 13 lines.
  * Reduce page 6, column 2 by 13 lines.
  * Reduce page 7, column 1 by 13 lines.
  * Reduce page 7, column 2 by 13 lines.
  * Reduce page 8, column 1 by 13 lines.
  * Reduce page 8, column 2 by 13 lines.
  * Reduce page 9, column 1 by 12 lines.
  * Reduce page 9, column 2 by 12 lines.
  * Reduce page 10, column 1 by 12 lines.
```

```
Note: 12 lines = 2.4" = 6.1 cm = 26.8% of the column height
```

If you're curious how the document managed to shrink substantially as the result of a relatively minor text reduction, you can typeset the document with a particular page and column enlarged by a given amount:

```
wheretotrim --debug=9,1,12 latex myfile.tex
```

2.3 Caveats

`wheretotrim` hooks into LaTeX's output routines, which are notoriously arcane and somewhat fragile. As a result, it is quite likely that `wheretotrim` will fail to analyze a large set of documents. Use the **--verbose** flag to help identify any problems that `latex` encounters.

In many cases, `wheretotrim` will recover by simply ignoring a few possible page and column expansions. For example, certain expansions may result in a `Float(s) lost` message. In other cases, `wheretotrim` will fail to analyze any modification to the document. For example, it may receive an `Infinite glue shrinkage found in box being split` error from every page and column variation it tries. In this particular case, see the discussion at `http://www.michaelshell.org/tex/ieeetran/`.

When `wheretotrim` is used with a `latex` auto-build script you may need to take measures to force the script to rebuild the document even if it appears that no files have changed. For example, `latexmk` should be given the `-CF` option to force rebuilding:

```
wheretotrim latexmk -CF myfile.tex
```

2.4 Restrictions

`wheretotrim` is implemented as a Perl script with an auxiliary LaTeX 2_ε package. It has been tested only on Linux, but I suspect that it should also work on OS X. I doubt it will work under Windows, though, due to the way the script uses a `bash`-specific technique for redirecting the standard error device into the standard output device.

3 Package implementation

This section presents the commented LaTeX source code for the wheretotrim package. Read this section if you want to learn how the package is implemented. Note that the package is not intended to be used explicitly (i.e., via `\usepackage`) but rather implicitly by the `wheretotrim` Perl script.

3.1 Theory of operation

The wheretotrim package mimics the effect of reducing a given page and column of a document by a given number of line heights. For simplicity, it does so by enlarging the specified column (so as to fit additional lines of text) rather than by reducing the amount of text in that column.

Users are not expected to load the wheretotrim package explicitly. Instead, whenever the `wheretotrim` script needs to observe the effect of enlarging a given column, it creates a temporary `.tex` file using the following template:

```
\RequirePackage[column=⟨absolute column⟩,
                expansion=⟨lines⟩,
                extracols=⟨padding columns⟩]{wheretotrim}
\PassOptionsToPackage{draft}{hyperref}
\input{⟨filename⟩}
```

where ⟨*absolute column*⟩ is the absolute column number to expand (with the first column on the first page being numbered 1); ⟨*lines*⟩ is the number of line heights (multiples of `\baselineskip`) by which to enlarge that column; ⟨*padding columns*⟩ is the number of extra full columns to append to the document (cf. Section 3.4); and ⟨*filename*⟩ is the name of the user's top-level LaTeX file.

The wheretotrim package works by modifying various TeX- and LaTeX-internal commands. At every `\shipout`, wheretotrim increases the absolute page counter. Whenever LaTeX constructs a column using `\@makecol`, wheretotrim logs the current absolute page and column numbers and invokes LaTeX's `\enlargethispage` macro when on the target page and column number. Because `\@makecol` is not called for every column, wheretotrim additionally modifies `\clearpage` and `\maketitle` also to conditionally enlarge the current column.

At the end of the document, wheretotrim outputs `\baselineskip` and `\textheight`, as these are needed by the `wheretotrim` script.

3.2 Package options

The wheretotrim package accepts three package options—`column`, `expansion`, and `extracols`—which are described below in the context of, respectively, `\wtt@target@column`, `\wtt@column@expand`, and `\wtt@extra@full@columns`.

`\wtt@target@column` `\wtt@target@column` is set by the `column` option and defaults to nonexistent column 0. It specifies the absolute column number to expand.

```
1 \newcommand{\wtt@target@column}{0}
```

`\wtt@column@expand` The `\wtt@column@expand` length—implemented as an ordinary macro—is set by the `expansion` option and defaults to 0 pt. It specifies the number of lines by which to expand that column (i.e., the multiple of `\baselineskip`).

```
2 \newcommand{\wtt@column@expand}{0pt}
```

\wtt@extra@full@columns

`\wtt@extra@full@columns` is set by the `extracols` option and specifies the number of additional, dummy, full columns to append to the end of the document to force spillover onto an additional page.

```
\newcommand{\wtt@extra@full@columns}{0}
```

We use the keyval package to help with option processing as it's widely available and wheretotrim's option-processing needs are fairly simple.

```
\RequirePackage{keyval}
\define@key{wtt}{column}{\gdef\wtt@target@column{#1}}
\define@key{wtt}{expansion}{%
  \xdef\wtt@column@expand{#1\noexpand\baselineskip}%
}
\define@key{wtt}{extracols}{\gdef\wtt@extra@full@columns{#1}}
```

\next

Process our options. We need to expand `\CurrentOption` before passing it to keyval's `\setkeys` macro, however.

```
\DeclareOption*{%
  \edef\next{\noexpand\setkeys{wtt}{\CurrentOption}}%
  \next
}
\ProcessOptions\relax
```

3.3 Column enlargement

\c@wtt@true@page

The wheretotrim package needs to keep track of the current page number. The page counter is unsuitable for this task because it is really a page *name*. That is, (1) it is not necessarily numeric (e.g., it may be a roman numeral while in the document's front matter), and (2) it is not necessarily unique (e.g., page may be 1 on the title page, abstract, and first page of text). To address this limitation we define a `wtt@true@page` counter and, with the help of the everyshi package, prepare for it to be incremented on every TEX page shipout.

```
\newcounter{wtt@true@page}
\setcounter{wtt@true@page}{1}

\RequirePackage{everyshi}
\EveryShipout{\addtocounter{wtt@true@page}{1}}
```

\c@wtt@column@num

The wheretotrim package also needs to keep track of the current absolute column number. By "absolute" we mean a running column number that does not reset to 1 on each page. We define a `wtt@column@num` counter to hold the current column number, and, below, we modify LATEX 2ε's `\@makecol` macro to increment it. Note that some pages may contain fewer pages than others due to, for example, `\clearpage` calls that cause pages to ship out early.

```
\newcounter{wtt@column@num}
```

\wtt@makecol

Before redefining `\@makecol`, we store its old definition in `\wtt@makecol`.

```
\let\wtt@makecol=\@makecol
```

\@makecol

\@makecol is LaTeX 2ε's primary mechanism for typesetting a column:

> \@makecol: Makes the contents of \box255 plus the accumulated footnotes, plus the floats in \@toplist and \@botlist, into a single column of height \@colht (unless the page height has been locally changed), which it puts into box \@outputbox. It puts boxes in \@midlist back onto \@freelist and restores \maxdepth.

Here, we augment \@makecol with code to report the current column and page number—and for the user's convenience, page name (\thepage). Our redefined \@makecol then increments the current absolute column number and compares it against \wtt@target@column. If equal, it uses LaTeX 2ε's \enlargethispage to increase the column height. Finally, it invokes the original \@makecol (stored in \wtt@makecol to typeset the column.

```
\def\@makecol{%
  \PackageInfo{wheretotrim}%
              {Column \thewtt@column@num\space is on page
               \thewtt@true@page\space (\thepage)}%
  \addtocounter{wtt@column@num}{1}%
  \ifnum\value{wtt@column@num}=\wtt@target@column
    \enlargethispage{\wtt@column@expand}%
  \fi
  \wtt@makecol
}
```

\wtt@clearpage

Before redefining \clearpage, we store its old definition in \wtt@clearpage.

```
\let\wtt@clearpage=\clearpage
```

\clearpage

LaTeX 2ε's \clearpage macro frustrates the wheretotrim package's attempts to enlarge a given column. We therefore redefine \clearpage first to perform its ordinary behavior (stored in \wtt@clearpage, then to check the value of the preceding penalty item. If the last penalty is -10001 then this is an opportune time to insert an \enlargethispage (assuming the current column is equal to \wtt@target@column). Because \clearpage may be called multiple times in a row and may be followed by \@makecol we restore the column counter to its prior value after comparing it to \wtt@target@column so it is not multiply incremented.

Note that \cleardoublepage internally calls \clearpage so it is sufficient to redefine only \clearpage.

```
\gdef\clearpage{%
  \wtt@clearpage
  \ifnum\lastpenalty=-10001\relax
    \addtocounter{wtt@column@num}{1}%
    \ifnum\value{wtt@column@num}=\wtt@target@column
      \enlargethispage{\wtt@column@expand}%
    \fi
    \addtocounter{wtt@column@num}{-1}%
  \fi
}
```

Wait until after the `\begin{document}` to redefine `\maketitle` in case `\maketitle` is modified before that point.

```
\AtBeginDocument{%
```

`\wtt@maketitle` Before redefining `\maketitle`, we store its old definition in `\wtt@maketitle`.

```
  \let\wtt@maketitle=\maketitle
```

`\maketitle` `\maketitle` is problematic macro for the wheretotrim package because of the way it switches into two-column mode within a one-column document (via LaTeX 2ε's `\twocolumn` macro). For lack of a more general solution we redefine `\maketitle` to enlarge the column only after typesetting the title and only when in two-column mode. Otherwise, the `\enlargethispage` inserted by `\@makecol` already had its intended effect.

```
  \gdef\maketitle{%
    \wtt@maketitle
    \if@twocolumn
      \ifnum\value{wtt@column@num}=\wtt@target@column
        \enlargethispage{\wtt@column@expand}%
      \fi
    \fi
  }%
}
```

3.4 Page spillover

Normally, it would not be possible to reduce page count by enlarging the last column by any amount. The trick we use here is to add to the end of the document a full column or two to make the document spill over onto an additional page, as illustrated by Figure 1. Thus, enlarging the last column by the height of the text it contains will enable a padding column to shift into that column and reduce the page count.

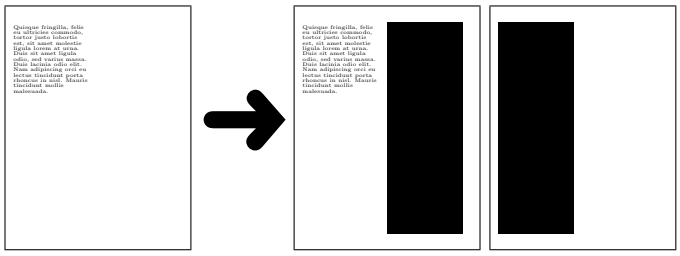

Figure 1: Padding a document with extra columns to induce page spillover

```
\AtEndDocument{%
```

Add zero, one, or two columns of padding (a `\parbox` of width `\linewidth` and height `\textheight`) based on the value of `\wtt@extra@full@columns` (set by the `extracols` package option).

```
\ifnum\wtt@extra@full@columns>0\relax
  \noindent\parbox[t][\textheight]{\linewidth}{%
    \rule{\linewidth}{\baselineskip}}\par
  \ifnum\wtt@extra@full@columns>1\relax
    \noindent\parbox[t][\textheight]{\linewidth}{%
      \rule{\linewidth}{\baselineskip}}\par
  \fi
\fi
```

Also at the end of the document, output the value of `\baselineskip` and the value of `\textheight`, as these are read by the `wheretotrim` script.

```
\PackageInfo{wheretotrim}%
            {Baseline skip: \the\baselineskip}%
\PackageInfo{wheretotrim}%
            {Text height: \the\textheight}%
}
```

4 Script implementation

This section presents the commented LaTeX source code for the `wheretotrim` Perl script. Read this section if you want to learn how the script is implemented.

4.1 Subroutine defintions

```
#! /usr/bin/env perl
```

```
use File::Basename;
use File::Temp qw(tempfile);
use Getopt::Long;
use POSIX;
use Pod::Usage;
use warnings;
use strict;
```

Define some global variables.

```
my $progname = basename $0;   # Name of this program
my $logfile;                  # LaTeX-generated log file
my $verbosity = 1;            # Level of output verbosity
my $allpages = 0;             # 1=report changes needed for all pages; 0=any page
my @latexcmd;                 # Complete command to run LaTeX
my $ltxfile;                  # Name of input file
my $colsperpage = 1;          # Number of columns per page (1 or 2)
my %column2page;              # Map from absolute column number to {page, column}
my $debugexp;                 # Typeset using an expansion of <page>,<column>,<expansion lines>
our $VERSION = "1.0";         # Version number of this program
```

`basename_newsuffix` Define a subroutine that replaces a file name with its base name and (optionally) new suffix.

```
sub basename_newsuffix ($;$)
{
```

```
    my ($fname, $newsuffix) = @_;
    my ($basename, undef, undef) = fileparse($fname, qr/.[^.]*/);
    $newsuffix = "" if !defined $newsuffix;
    return $basename . $newsuffix;
}
```

`create_latex_file` Define a subroutine to create a temporary LaTeX file that modifies a few LaTeX commands then loads the user's document. The subroutine returns the name of the temporary file.

```
sub create_latex_file ($$$)
{
    my ($columntoexpand, $columnexpandlines, $extrafullcolumns) = @_;
    my ($modltx, $modltxfile) = tempfile("wtt-XXXXXX",
                                         TMPDIR => 1,
                                         SUFFIX => ".tex",
                                         UNLINK => 1);
    print $modltx "RequirePackage[column=$columntoexpand,expansion=$columnexpandlines,extracols
    print $modltx "PassOptionsToPackage{draft}{hyperref}n";   # Avoid "pdfendlink ended up in d
    print $modltx "input{$ARGV[$#ARGV]}n";
    close $modltx;
    return $modltxfile;
}
```

`run_latex` Define a subroutine to run LaTeX on a given filename.

```
sub run_latex ($$$$)
{
```

Add some additional arguments to the LaTeX command.

```
    my ($modltxfile, $columntoexpand, $columnexpandlines, $extrafullcolumns) = @_;
    my $jobname = basename_newsuffix($ltxfile);
    @latexcmd = (@ARGV[0..$#ARGV-1], "-jobname=$jobname", $modltxfile);
```

Run LaTeX.

```
    if ($verbosity == 1) {
        if ($columntoexpand == 0) {
            print "Compiling $ltxfile normally";
            if ($extrafullcolumns > 0) {
                printf ", but with %s column%s of padding", $extrafullcolumns, $extrafullcolumn
            }
            print " ... ";
        }
        elsif ($colsperpage == 1) {
            my ($page, $col) = @{$column2page{$columntoexpand}};
            printf "Compiling %s with page %d expanded by %d line%s ... ",
            $ltxfile, $page, $columnexpandlines, $columnexpandlines == 1 ? "" : "s";
        }
        else {
            my ($page, $col) = @{$column2page{$columntoexpand}};
            printf "Compiling %s with page %d, column %d expanded by %d line%s ... ",
            $ltxfile, $page, $col, $columnexpandlines, $columnexpandlines == 1 ? "" : "s";
```

```
127         }
128     }
129     elsif ($verbosity > 1) {
130         print "Running @latexcmd\n";
131     }
132     open(LATEX, "-|", "sh", "-c", 'echo X | "$@" 2>&1', "--", $latexcmd[0], @latexcmd[1..$#late
133     while (my $oneline = <LATEX>) {
134         print $oneline if $verbosity > 1;
135     }
136     close LATEX;
137     my $errcode = $?;
138     if ($verbosity == 1) {
139         print $errcode == 0 ? "done.\n" : "failed.\n";
140     }
141     elsif ($verbosity > 1) {
142         print "Finished running.\n";
143     }
144     return $errcode;
145 }
```

`process_log_file` Define a subroutine to process a log file and return various data extracted from it.

```
146 sub process_log_file ($$$)
147 {
148     my ($columntoexpand, $columnexpandlines, $extrafullcolumns) = @_;
149     my %column_map;
```

Extract wheretotrim information lines and the final page count.

```
150     print "Processing $logfile ... " if $verbosity > 0;
151     my ($numpages, $baselineskip, $textheight) = (0, 0, 0);
152     open(LOGFILE, "<", $logfile) || die "${progname}: Failed to open $logfile ($!)\n";
153     my $infostr = "Package wheretotrim Info";
154     while (my $oneline = <LOGFILE>) {
155         $baselineskip = $1+0 if $oneline =~ /^$infostr: Baseline skip: ([\d.]+)pt/;
156         $textheight = $1+0 if $oneline =~ /^$infostr: Text height: ([\d.]+)pt/;
157         $column_map{$1} = [$2, $3] if $oneline =~ /^$infostr: Column (\d+) is on page (\d+) \((
158         $numpages = $1 if $oneline =~ /^Output written on.*\((\d+) page/;
159     }
160     close LOGFILE;
161     $numpages-- if $extrafullcolumns > 0;
162     if ($verbosity > 0) {
163         printf "done (%d page%s).\n",
164         $numpages, $numpages == 1 ? "" : "s",
165     }
166     return ($numpages, $baselineskip, $textheight, \%column_map);
167 }
```

`latex_page_count` Define a subroutine to run LaTeX and return a page count and other information.

```
168 sub latex_page_count ($$$)
169 {
```

LaTeX wrapper scripts might not like being given LaTeX code on the command line. We therefore create a temporary file that prepares LaTeX for programmatically modifying column heights.

```
    my ($columntoexpand, $columnexpandlines, $extrafullcolumns) = @_;
    my $modltxfile = create_latex_file($columntoexpand, $columnexpandlines, $extrafullcolumns);
```

Run latex on the temporary file.

```
    my $errcode = run_latex($modltxfile, $columntoexpand, $columnexpandlines, $extrafullcolumns
    unlink $modltxfile;
```

Process the log file.

```
    return (0, undef, undef, undef) if $errcode != 0;
    return process_log_file($columntoexpand, $columnexpandlines, $extrafullcolumns);
}
```

4.2 Main program execution

Parse the command line.

```
my $wanthelp = 0;
my $wantversion = 0;
Getopt::Long::Configure("require_order");
GetOptions("h|help"       => \$wanthelp,
           "V|version"    => \$wantversion,
           "a|allpages"   => \$allpages,
           "l|logfile=s"  => \$logfile,
           "v|verbose+"   => \$verbosity,
           "d|debug=s"    => \$debugexp,
           "q|quiet"      => sub {$verbosity = 0})
    || pod2usage(-exitval => 2);
if ($wantversion) {
    print "wheretotrim $VERSION\n";
    exit 0;
}
pod2usage(-verbose => $verbosity,
          -exitval => 1) if $wanthelp;
pod2usage(-message => "${progname}: A latex command must be specified",
          -exitval => 2) if $#ARGV == -1;
$ltxfile = basename($ARGV[$#ARGV]);
$logfile = basename_newsuffix($ARGV[$#ARGV], ".log") if !defined $logfile;
```

Determine the document's baseline characteristics.

```
my ($basepages, $baselineskip, $textheight, $c2p_ptr) = latex_page_count 0, 0, 0;
die "${progname}: Failed to build $ltxfile\n" if $basepages == 0;
%column2page = %$c2p_ptr;
print "\n" if $verbosity > 0;
```

Map an absolute column to a page and column number.

```
my $prevpage = 0;
foreach my $col (sort {$a <=> $b} keys %column2page) {
    my ($pagenum, $pagename) = @{$column2page{$col}};
```

```
    if ($pagenum == $prevpage) {
        $column2page{$col} = [$pagenum, 2, $pagename];
        $colsperpage = 2;
    }
    else {
        $column2page{$col} = [$pagenum, 1, $pagename];
    }
    $prevpage = $pagenum;
}
```

If we were given a page, column, and expansion, typeset the document with those parameters and exit.

```
if (defined $debugexp) {
    die "${progname}: Failed to parse \"$debugexp\" into {page, column, expansion}\n" if $debug
```

Convert page and column number to absolute column number.

```
    my ($target_page, $target_col, $expansion) = ($1, $2, $3);
    my $testcol;
    while (my ($abscol, $page_col) = each %column2page) {
        if ($target_page == $page_col->[0] && $target_col == $page_col->[1]) {
            $testcol = $abscol;
            last;
        }
    }
    die "${progname}: Failed to map page $target_page, column $target_col to an absolute column
```

Enlarge the given page.

```
    my ($numpages, undef) = latex_page_count $testcol, $expansion, $colsperpage;
    print "\n" if $verbosity > 0;
    latex_page_count $testcol, $expansion, 0;   # Run again without appending any extra columns
    print "\n" if $verbosity > 0;
    print "Expanding page $target_page, column $target_col by $expansion lines ";
    if ($numpages == $basepages) {
        print "does not reduce the page count below $numpages pages.\n";
    }
    else {
        print "reduces the page count from $basepages pages to $numpages pages.\n";
    }
    exit 0;
}
```

Determine columns for which no amount of expansion will reduce the page count.

```
my $maxexpansion = int($textheight/$baselineskip + 1);
my @complete = (0, 0+keys %column2page);   # Fraction complete (numerator and denominator)
foreach my $expcol (sort {$a <=> $b} keys %column2page) {
    my ($numpages, undef) = latex_page_count $expcol, $maxexpansion, $colsperpage;
    if ($verbosity > 0) {
        $complete[0]++;
        printf "Trial runs are %.1f%% complete.\n\n", 100.0*$complete[0]/$complete[1];
    }
```

```
246     delete $column2page{$expcol} if $numpages > 0 && $numpages == $basepages;
247 }
```

Keep expanding a page by greater and greater amounts until we reduce our page count.

```
248 my %col2savings;     # Map from an absolute column to an {expansion, page count} tuple.
249 my $target_num_cols = $allpages ? (keys %column2page) : 1;   # Minimum number of columns for wh
250 my $minexpansion;    # Minimum value of the above that saves a page
251 @complete = (0, $maxexpansion*keys %column2page);
252 foreach my $expansion (1 .. $maxexpansion) {
```

Expand each column in turn.

```
253     foreach my $expcol (sort {$a <=> $b} keys %column2page) {
254         $complete[0]++;
255         next if defined $col2savings{$expcol};   # Already finished
256         next if $column2page{$expcol}->[0] == $basepages && $column2page{$expcol}->[1] == 2;
257         my ($numpages, undef) = latex_page_count $expcol, $expansion, $colsperpage;
258         if ($numpages > 0 && $numpages < $basepages) {
259             $col2savings{$expcol} = [$expansion, $numpages];
260             $minexpansion = $expansion if !defined $minexpansion;
261         }
262         if ($verbosity > 0) {
263             printf "Execution is %.1f%% complete.\n\n", 100.0*$complete[0]/$complete[1];
264         }
265     }
266     last if keys %col2savings >= $target_num_cols;    # Success
267 }
```

Restore the document to its original form.

```
268 run_latex $ltxfile, 0, 0, 0;
269 printf "Execution is 100.0%% complete.\n\n" if $verbosity > 0;
```

Output the space savings.

```
270 if (keys %col2savings == 0) {
271     printf "It does not appear possible to reduce the page count from %d to %d\n",
272     $basepages, $basepages-1;
273     print "by removing any amount of text from any single column.\n\n";
274     exit 0;
275 }
276 printf "To reduce the page count from %d to %d, do %s following:\n\n",
277     $basepages, $basepages-1, keys %col2savings == 1 ? "the" : "any of the";
278 foreach my $abscol (sort {$a <=> $b} keys %col2savings) {
279     my ($expansion, $numpages) = @{$col2savings{$abscol}};
280     my ($page, $col, $pagename) = @{$column2page{$abscol}};
281     print "  * Reduce page $page";
282     print " (\"$pagename\")" if $pagename ne $page;
283     print ", column $col" if $colsperpage > 1;
284     printf " by %d %s", $expansion, $expansion == 1 ? "line" : "lines";
285     if ($numpages < $basepages - 1) {
286         printf " (produces %d %s)", $numpages, $numpages == 1 ? "page" : "pages";
287     }
```

```
288     print ".\n";
289 }
290 print "\n";
291 my $minpoints = $minexpansion*$baselineskip;
292 printf "Note: %d lines = %.1f\" = %.1f cm = %.1f%% of the %s height\n",
293     $minexpansion, $minpoints/72.27, $minpoints/28.45,
294     100.0*$minpoints/$textheight, $colsperpage == 1 ? "page" : "column";
```

The `wheretotrim` script ends with POD-format documentation. This is not listed here because it is largely redundant with the contents of Sections 1 and 2. See those sections for documentation about `wheretotrim`'s usage.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Q

R

S

T

V

W