Glisterings

Peter Wilson

> His eye, which scornfully glisters like fire,
> Shows his hot courage and his high desire.
>
> *Venus and Adonis*, William Shakespeare

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Corrections, suggestions, and contributions will always be welcome.

> Words are wise men's counters, they do but reckon with them, but they are the money of fools.
>
> *Leviathan*, Thomas Hobbes

1 Counting

1.1 Number of words

Some publications put word limits on manuscripts, and the question often arises as what is the (best) way to count them. This is answered in detail in the FAQ [1] but my answer is to simply count the number of words on one page of your manuscript and multiply by the number of pages. This is essentially the technique used by book designers and publishers when confronted with a manuscript in the process called *casting off* (see, for example, [3, Chap. 8]). The publishers are not particularly interested in the exact number of words but are very much interested in the number of pages in the final production.

If you are writing a thesis the powers-that-be may specify a word limit, but it is probably safe to assume that they will not actually check the number of words themselves, unless there is an obvious mismatch between your number and the size of the thesis. In any event, what counts as a 'word'? Is 'powers-that-be' one word or three? How many 'words' are the equivalent of a table or a figure? If different sized fonts will be used, are all 'words' equal? What about footnotes, mathematical equations, verse — how many 'words' should be allocated to them?

1.2 Lua

By the time you read this LuaTEX should be available, and perhaps you have used it. At the time of writing it is still being developed and I have not tried it. However, assuming that Lua [2] will be available on all TEX platforms, I thought that I would try and use it for its own sake.

I have been fortunate in being able to use lead type and a hand operated press, much as Gutenberg did in the 15th century. Unlike digital typesetting where you can have an unlimited number of characters of any particular kind, the number of available characters is strictly limited — if the font you are using has only 23 'e' sorts (a *sort* is a single piece of lead type), then in one go you can only set text that has no more than 23 'e' characters. It is therefore important to know how many of each sort is required to set a page of text. For example, I wanted to print a 16th century poem — only 2 verses on one page — in a particular font but I couldn't do so as I was one 'h' short (there were a lot of thee, thou, thine, ... eth, etc., words compared to modern English). I work on a Linux system which provides programs for counting the number of words and the total number of characters in a piece of text; presumably other systems provide the equivalent. But what I wanted was a program to count the numbers of the individual characters — the number of 'A' characters, the number of 'a' characters, and so on.

I managed to extend a Lua program that would do this for me. Here it is, in a file that is called `gwc.lua`:

```
#!/usr/local/bin/lua5.1
-- gwc.lua Lua program to count characters, etc
-- (see Lua Manual p.198)
-- call as: gwc.lua file

local BUFSIZE = 2^13        -- 8k
local f = io.input(arg[1])  -- open input file
local cc = 0       -- count of chars
local lc = 0       -- count of lines
local wc = 0       -- count of words
local ct = {}      -- table of char counts
local k, v         -- table key and value
for i = 32,126 do  -- initialise ASCII slots
  ct[i] = 0
end
local T = 0        -- my total chars
local tc = 0       -- actual total chars
                   -- (no newlines, etc)
while true do
  -- read a chunk of text
  local lines, rest = f:read(BUFSIZE, "*line")
  if not lines then break end
  if rest then
     lines = lines .. rest  .. "\n" end
  cc = cc + #lines
  -- count words in the chunk
  local _, t = string.gsub(lines, "%S+", "")
  wc = wc + t
  -- count newlines in the chunk
  _, t = string.gsub(lines, "\n", "\n")
  lc = lc + t
```

```
-- make a list of character frequencies
  local K
  for i = 1, string.len(lines) do
    K = string.byte(lines,i)
    if K > 32 then
     if K < 126 then
      ct[K] =  ct[K] + 1
      T = T + 1
     end
   end
  end
end

-- strip off input (e.g., fin.ext) file's
-- extension and make output file fin.gwc
base, ext = string.match(arg[1],
                  "(%w+)%.(%w+)")
ofile = base..".gwc"

-- cc includes newlines, so T = (lc + wc)
tc = cc - lc - wc
io.output(ofile)
io.write("Character counts in file ",
         arg[1], "\n")
io.write("", "lines =", lc, "\n",
             "words =", wc, "\n",
             "characters = ", tc, "\n\n")

io.write("Character total\n")
for k,v in pairs(ct) do
  if v > 0 then
    print(string.char(k),v)
    io.write("  ", string.char(k), "      ",
             string.format("%4d",v), "\n")
  end
end
print("Output saved in: ", ofile)
```

That ends the Lua program. In this case a 'word' is a sequence of characters followed by one or more spaces. I was only interested in characters corresponding to the sorts in the fonts that were available to me. Being English this fortunately restricted the characters to the ASCII printable character set. If you need to count other characters then you will have to extend the program. The Lua manual [2, p. 198] describes how the word and line count part of the program works in more detail.

> Change is not made without inconvenience, even from worse to better.
>
> *A Dictionary of the English Language: Preface*, Samuel Johnson

2 Changing the layout

A question that pops up from time to time is 'How do I change the layout for a particular page?', where the 'layout' includes items like the size and location of the textblock, and different headers and footers.

2.1 The shape of the page

You can do many things, but one that you cannot do is to change the textwidth in the middle of a paragraph. For instance if the textblock is 30pc wide on one page and 25pc wide on the following page, then a paragraph that starts on the first page and continues onto the next will be 30pc wide on both pages. This is because TEX internally typesets paragraph by paragraph according to the current textwidth. Having set a paragraph it then decides if there should be a pagebreak in it. If there is it puts the beginning of the already laid out paragraph on the first page and the remainder, which is already set internally, goes on the following page(s) with the *same* textwidth.

The general page layout parameters are diagrammed in Figure 1.

To change the height of the textblock on a particular page, the LATEX `\enlargethispage` macro can be used. This takes a single length argument which is added to the textheight for the page on which it occurs — a positive length increases the textheight and a negative one decreases it. The change is made at the bottom of the textblock; the location of the top of the textblock is unchanged.

The `quote` and `quotation` environments temporarily change the margins and width of the textblock, and you can do the same by using, for example, the `adjustwidth` environment provided by the `changepage` package [5].

> The `adjustwidth` environment takes two length arguments, and increases the left and right margins by the given amounts. For example, I used
> `\begin{adjustwidth}{3em}{1.5em}`
> at the start of this paragraph, and will end `adjustwidth` at the end of the paragraph.

The page layout parameters used are those in effect at the start of a page when the first item (e.g., a character, a box, etc.) is put onto the page. Layout changes after that will not be effective until the start of the next page. You can, though, change the text width *between* pages. The trick here is that when you change from one column to two columns, or vice versa, LATEX recalculates its view of the layout. The general scheme is to clear the page, change the layout parameters, then set the number of columns which starts the same new page again but with the layout changes implemented. Assuming a one column document, the general procedure is:

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.

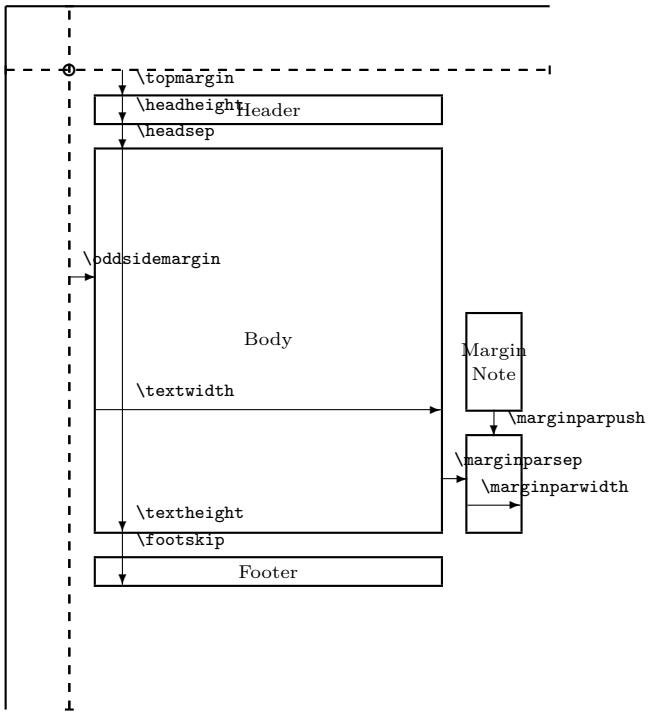

Figure 1: LaTeX page layout parameters for a recto page

```
\clearpage
% change textblock, margins, ...
\onecolumn
```

If you need them, the changepage package provides macros to '`change textblock, margins, ...`'.

Just so you can see what happens, the kernel definition of `\onecolumn` is:

```
\def\onecolumn{%
  \clearpage
  \global\columnwidth\textwidth
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumnfalse
  \col@number \@ne
  \@floatplacement}
```

The code for `\twocolumn` is similar but does a little more, especially as it takes an optional argument although that has no effect on the various width settings.

As an example, if you needed to have different text heights and widths for one set of pages, those in the frontmatter perhaps, than for another set, say the rest of the work, you could define

```
\newcommand*{\addtotextheightwidth}[2]{%
  \clearpage
  \addtolength{\textheight}{#1}
  \addtolength{\textwidth}{#2}
  \onecolumn}
```

and use it when you need to make a change.

2.2 Headers and footers

Another kind of layout change that I have seen requested is to add 'Page' above the page numbers in the Table of Contents or List of Figures, etc. As an example say that the requirement is that for the List of Figures (LoF) the word 'Figure' should be placed flushleft at the start of the column of figure titles and the word 'Page' flushright above the page numbers; if the LoF continues for more than one page, these should be repeated at the start of each page. The page number(s) of the LoF itself should be centered at the bottom of the page (i.e., the plain pagestyle). There are similar requirements for the Table of Contents (ToC) and List of Tables (LoT), but I'll just show how the LoF requirements can be met.

Changing pagestyles can be accomplished with the fancyhdr package [4] but I will assume that the memoir class [6] is being used which includes similar facilities.

The memoir class lets you define as many pagestyles as you want. We need a pagestyle for any

LoF continuation pages (and others for the ToC and LoT). Here's the one for the LoF, which I have called the lof pagestyle. This puts the page number centered in the footer and 'Figure' at the left in the header and 'Page' at the right.

```
\makepagestyle{lof}% a new pagestyle
  \makevenfoot{lof}{}{\thepage}{} % like plain
  \makeoddfoot{lof}{}{\thepage}{} % like plain
  \makeevenhead{lof}{Figure}{}{Page}
  \makeoddhead{lof}{Figure}{}{Page}
```

When we start the LoF we need to make sure that the lof pagestyle will be used for any continuation pages. We can do this by adding the necessary code to the `\listoffigures` command, and memoir provides the `\addtodef` command for doing this. It takes three arguments, the first is the name of a macro, the second is code to be added at the start of the macro's definition and the third is code to be added at the end of the macro's definition.

```
\addtodef{\listoffigures}{%
  \clearpage\pagestyle{lof}}{}
```

Memoir provides a command that is called before setting the title of the LoF and another that is called after the title. You can redefine these to do what you want. In this case we just need to extend what happens after the title.

```
\renewcommand*{\afterloftitle}{%
  \thispagestyle{plain}
  \par\nobreak
  {\normalfont\normalsize Figure \hfill Page}
  \par\nobreak}
```

The above makes the first page of the LoF use the plain pagestyle, and then puts a line containing 'Figure' at the left and 'Page' at the right. The actual listing of titles and page numbers will start after these preliminaries.

Setting up the ToC and LoT is almost identical to the above, but with names changed.

One thing to watch for is that after the LoF has been processed the lof pagestyle is still in effect. After the LoF has finished it will be necessary to revert back to the regular pagestyle which, for the sake of argument, let's say is heads. To be on the safe side the general scheme, then, is:

```
\documentclass[...]{memoir}
%% define heads pagestyle
%% ToC, LoF, ToC changes
%% more preamble
\addtodef{\mainmatter}{}{\pagestyle{heads}}
\pagestyle{heads}
\begin{document}
%% title pages
%% maybe Preface and such
%% \tableofcontents\clearpage\pagestyle{heads}
%% \listoffigures\clearpage\pagestyle{heads}
%% \listoftables\clearpage\pagestyle{heads}
%% other prelims
\mainmatter
...
\end{document}
```

which ensures that at the start of the main matter the regular heads pagestyle is in effect, no matter what games were played beforehand.

References

[1] Robin Fairbairns. The UK TEX FAQ. Available on CTAN in `help/uk-tex-faq`.

[2] Roberto Ierusalimschy. *Programming in Lua, Second Edition*. Lua.org, Rio de Janeiro, 2006. ISBN 85-903798-2-5.

[3] Ruari McLean. *The Thames and Hudson Manual of Typography*. Thames and Hudson, 1980. ISBN 0-500-68022-1.

[4] Piet van Oostrum. Page layout in LATEX, 2004. Available on CTAN in `latex/macros/contrib/fancyhdr`.

[5] Peter Wilson. The changepage package, 2008. Available on CTAN in `latex/macros/contrib/misc/changepage.sty`.

[6] Peter Wilson. The memoir class for configurable typesetting, 2009. Available on CTAN in `latex/macros/contrib/memoir`.

◇ Peter Wilson
18912 8th Ave. SW
Normandy Park, WA 98166
USA
herries dot press (at)
earthlink dot net