How to Make a PDF File Into a Printable Book.

By Robert L Parker <rlp1938@gmail.com>

Introduction

There are many PDF files available both free and for sale that are entirely suitable for viewing on screen but they are unsuitable for printing on both sides of the paper and binding as a book because they have an equal margin all round, normally 25 mm in size. Many such publications also make liberal use of white text on coloured block backgrounds no doubt to discourage making the material into a book.

There are PDF GUI editors available particularly for Windows which are no doubt suitable for directly editing the PDF and so prepare it for printing and binding. I have taken another approach which uses all free software, some installable on Linux systems and some I have written myself. My experience is with Ubuntu which no doubt will translate directly for the other 'buntus, Debian and Mint. Users of RPM based systems may have to search their repositories to find the required programs. Moreover the margin adjustment is done at command line via scripts so it is much faster than in a GUI editor. However manual editing of postscript files is required for removing coloured boxes and so on.

Software Required

The programs we need are *pdftk, pstopdf, epstopdf* and *pstops*.
Get these by: 'apt-get install pdftk texlive-font-utils psutils' or use aptitude or synaptic if you prefer. Because you will need to compile a couple of helper programs which I have written you will also need to 'apt-get install build-essential manpages-dev'. The first package is absolutely required for compiling programs, the second if you want to understand the use of library calls within the programs. I will describe these helper programs later in this document. The programs are *odd moveup* and the scripts *burst2ps setmargins* and *makebook.* You can obtain these by downloading them from rlp1938.com/Software/pdf2book.current. Then unpack using 'tar -xzf pdf2book', cd into the pdf2book directory and then 'make && sudo make install'.

Get Started

Once you have all of the required software in place make a working directory named anything you like and copy your PDF file into it. Copy not move it, you can not be too safe with these operations. Open up a terminal and cd into the directory where you have copied your PDF file into, then: 'burst2ps yourbook.pdf'. If you then 'ls' in this directory you will then have many files named as pg_0001.ps, pg_0002.ps … to as many as there are pages in your document.
For more information 'man burst2ps'.

Put The Section or Chapter Headings on Right Facing Pages

The first page taken from the original PDF file is named pg_0001.ps and that is a right facing page. Consequently all odd numbered pages are right facing pages and the even numbers are left facing pages. By convention a new chapter or section is placed on a right facing page regardless of whether the preceding left page is entirely blank. If this operation is required you need a blank PDF single page file. It is very easy to produce, just open a new write document in OpenOffice/Libre Office and export the empty document as blank.pdf into the directory where you are working. If you are really anally retentive be free to put a heading on the page “This page is intentionally left blank”. Personally I wouldn't bother.

Then open your original PDF document in your pdf viewer and quickly hunt through for Chapter headings. For example suppose that “Chapter 1. ….” appears as page 14 in your PDF document, then open pg_0014.ps in your postscript viewer. (In Ubuntu the postscript and PDF viewer is the same program). Confirm then that pg_0014.ps is the page the introduces Chapter 1. If so then pg_0014.ps is required to be renamed to pg_0015.ps and likewise every file above that must also have it's name incremented. Do it like this:

'moveup pg_0014.ps' and then

'cp blank.pdf pg_0014.ps'

A better way is to:

'moveup -c blank.ps pg_0014.ps'

and *moveup* will automatically make the copy for you.

'man moveup' for more information.

Then repeat this procedure to the end of the book. NB if Chapter 2 happens to be at page 29 in the original PDF the applicable postscript page will be at pg_0030.ps not pg_0029.ps. Each time after a 'moveup' the postscript file will be another one further on from the page number in the original.

Alter the Margins on the Postscript Files

'setmargins right_page_left_margin_increase left_page_left_margin_decrease'

The parameters are the changes in points in the margin sizes for example:

'setmargins 36 36' because we want to increase the left margin on the right facing page and decrease the left margin on the left facing page.

You can optionally scale the image:

'setmargins -s 0.95 50 50'

The units used are points, a printers measure equal to 1/72 inch.

Setmargins creates the pagespecs for the program *pstops* and runs that program on each pg_????.ps file in turn,

For more information 'man odd' , 'man pstops' and 'man setmargins'.

Assemble the Book

'makebook' or 'makebook yourbookname.pdf'

The script makebook works on pg_*.ps, turns each in turn into a PDF using *epstopdf* with the same basename but with the extension .pdf, and then assembles it into one PDF file which you may optionally name or by default it will be named as book.pdf. The original postscript files are deleted as the PDF's are made and once the output PDF is made the source PDF's are deleted.

NB there is also a program pstopdf in the texlive-font-utils package. I have found that for some commercial PDF books this program shrinks the A4 PDF pages to the smaller format that I suppose is used for the commercial printed version of the book. The program epstopdf keeps the original A4 size so that is what I have used in this script.

See 'man epstopdf'.

Afterword

The program *pstops* along with the other programs that come in the *psutils* package is a very sophisticated piece of work. An experienced user may be able to simply convert the original PDF file to postscript using *pdftops* then use the modlulo operator in the pagespecs to select alternate pages with a different margin. That would save the burst and concatenate operations. However the method I show above is far simpler and it does leave the option of forcing chapter headings onto right facing pages.

What to do Next

Well if your newly made printable book.pdf contains no colour or you intend to print it on a colour printer there is nothing more to do except just print it. But if you want to print on say a mono laser printer and you have plenty of white print on large dark coloured backgrounds you may want change that to black text on a white background. If that is the case proceed as follows:

Convert your book.pdf to postscript

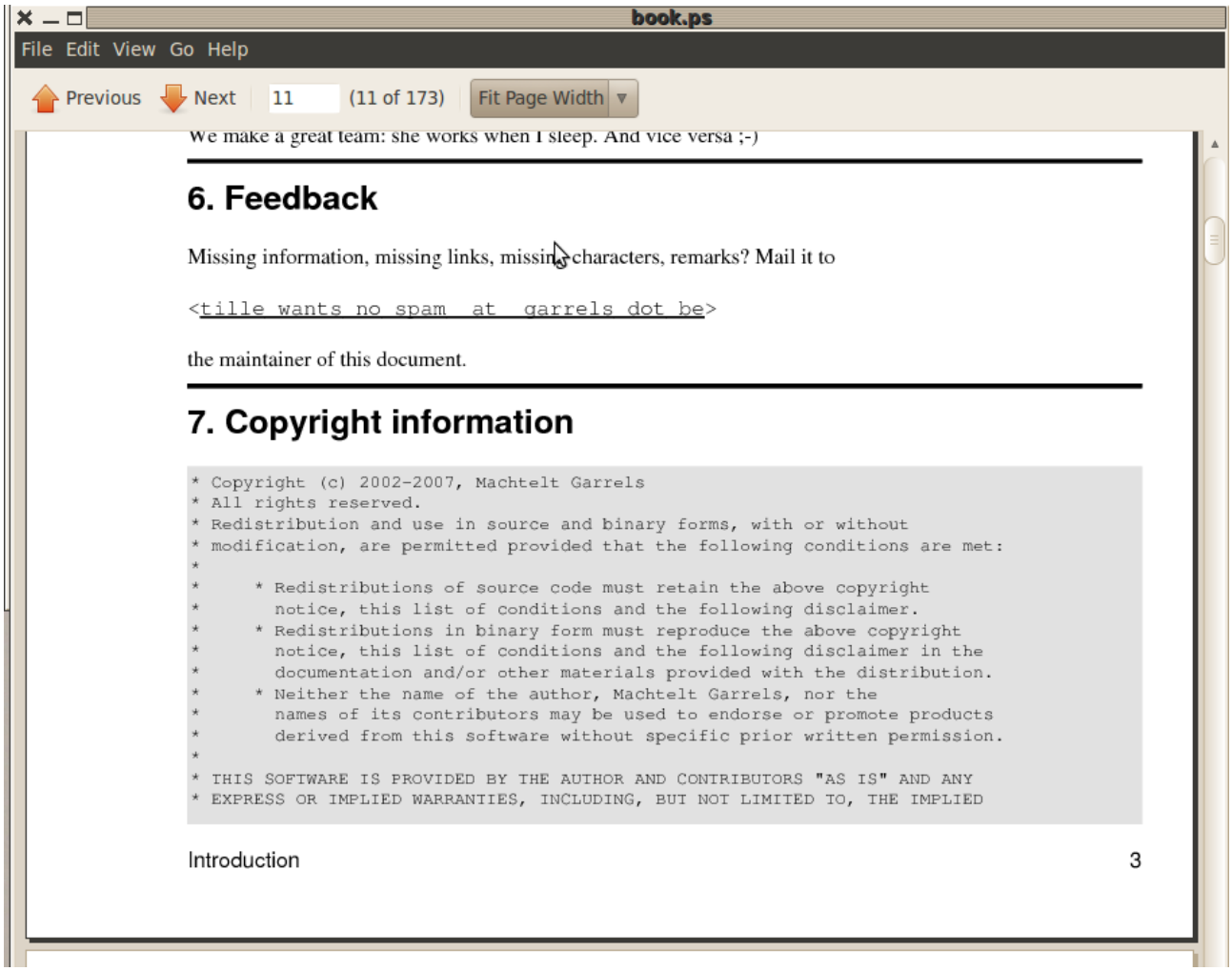
'pdftops book.pdf'

This will produce a file called book.ps. Open it in your document viewer and also in a text editor. I use Geany for editing this material and in what follows the screen shots will be taken from this editor. Scroll through book.ps until you find a splash of colour you want to alter. Note the page number in your viewer. In the text editor do a search for '%%Page:'. Repeat the search until your page number matches that in the viewer. NB The page number in a postscript file is repeated. Now if this enormous splash of colour happens to be an image then I cannot help you at this stage. Just rely on your laser printer to grayscale the stuff at print time. An image will appear in your postscript file something like this:

```
/DeviceRGB setcolorspace
<<
  /ImageType 1
  /Width 159
  /Height 250
  /ImageMatrix [159 0 0 -250 0 250]
  /BitsPerComponent 8
  /Decode [0 1 0 1 0 1]
  /DataSource currentfile
    /ASCII85Decode filter
    << >> /DCTDecode filter
>>
pdfIm
s4IA>!"M;*Ddm8XA,lT0!!3,S!(-_f!WiE)!WiE)!WrN+!sAf2"9S`/#6b;7"pkPA
#mgkC#RUqL$jm=N$4$nG&.B!X%hB3^#n.CY&J#Ka%hB3NgAjSA!s8W.!sAi3":#GE
…. Lots more of this gunk.
s5CMbB@gl%)[M%$j0/b7t?DA#&R/URE_?$HdlN=8O/79?TmYG?Uh^rnFWB90sX_TU
khrrC0$'"a21-ifnAJ-l9)!2'CD$32ISTE9WN!%7i?!e:UX-ifnAJ3X(e~>
%-EOD-
```

So let's forget about what we cannot change and move on to something we can.

Here is an example of what we can usefully change:



And now here is some the postscript which controls page 11 we see above. I have annotated it with my comments which begin with '#'. These comments are not part of the postscript file. Postscript comments begin with '%' and when doubled as '%%' they are a processor directive.

%%Page: 11 11 # This is the page number

%%BeginPageSetup

%%PageOrientation: Portrait

pdfStartPage

0 0 595 792 re W

%%EndPageSetup

[] 0 d

1 i

0 j

0 J

10 M

1 w

/DeviceGray {} cs

[0] sc

```
/DeviceGray {} CS
[0] SC
false op
false OP
{} settransfer
q
q
[0.1 0 0 0.1 0 0] cm
{} settransfer
q
100 0 5850 7920 re
W
/DeviceRGB {} cs              # this is the line we search for.
[0.8808 0.8808 0.8808] sc     # this line is the colour value as RGB light grey
820 580 4870 1824 re          # this line describes where to paint this colour
f
/DeviceRGB {} cs
[0 0 0] sc                    # this is the colour black, [1 1 1] is white.
820 6290 4870 20 re           # one of the 20 point ruled black lines
f
820 3954 4870 20 re           # another black line.
f
```

We should note at this point that there is no requirement for the data on a postscript page to flow from top to bottom. The elements may be positioned on the page in absolute units, or in relative units from where the last operation finished. In the section of the file shown above all of the painting of the black lines and light grey boxes is completed before any of the text is rendered.

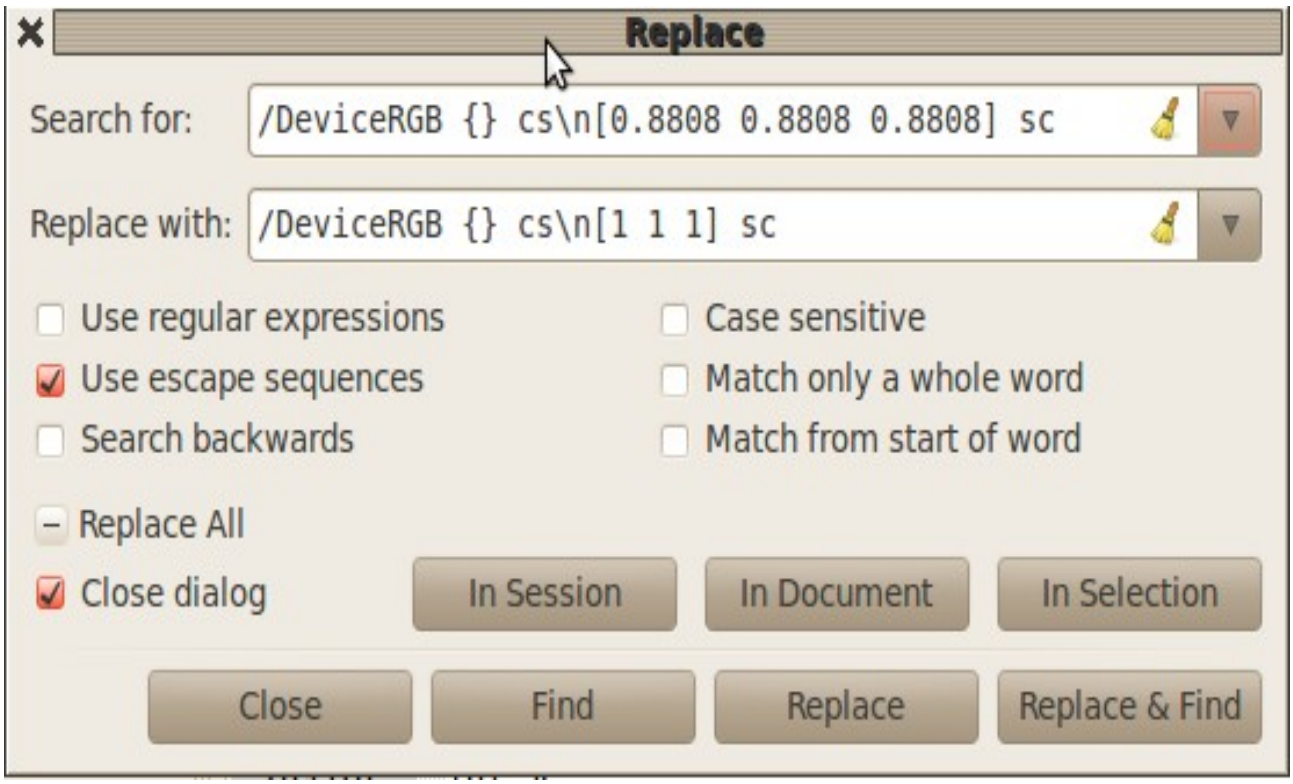
To edit a postscript file safely you can comment out an existing line and enter a new line below and end up with something like this:

```
/DeviceRGB {} cs              # this is the line we search for.
%[0.8808 0.8808 0.8808] sc    # this line is the colour value as RGB – light grey
                              # now commented out.
[1 1 1] sc                    # this line is the colour value as RGB – now white
820 580 4870 1824 re          # this line describes where to paint this colour
f
```

Once done save the file. The viewer is aware of changes to the file under view and will automatically reload the file in a second or so. If you enter invalid postscript into the file the viewer will simply stall with the message “Loading”. Revert the change and save the file again.

So once you are happy with the results of a change you can do a global replacement.

The screen shot below shows how to do it in Geany.



Do be very careful doing replacements on [1 1 1] or [0 0 0]. It's unlikely that a global replacements are possible with such. However coloured boxes, triangles and so on are fairly safe. Note though that any text in these objects may be white, [1 1 1] and will need to be changed to black, [0 0 0].

Once the editing is complete you can redo your PDF.

'epstopdf book.ps'

That will clobber your original PDF so alternatively:

'epstopdf –output=someothername.pdf book.ps'