Previous post

Next post

Automating the annotation of PDFs

December 4, 2021 at 3:34 PM by Dr. Drang

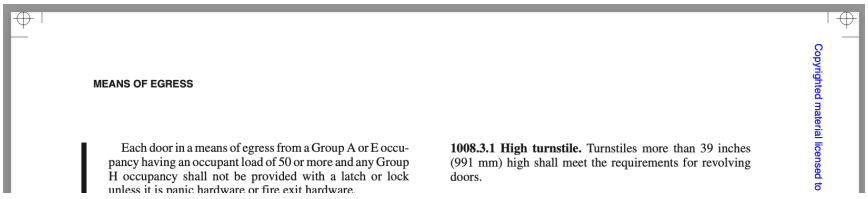
A few days ago, on the Automators podcast forum, thatchrisharper asked about a way to automatically add the filename to the first page of a PDF. While I knew of many tools that allow you to overlay one PDF on top of another, I didn't know of any to directly add text to a specific spot of a PDF. But it was the sort of thing I've occasionally needed to do, so I went looking for a solution. This post, most of which is in my answer to thatchrisharper's question, is what I found.

The solution comes from a combination of Ghostscript, which you can install through Homebrew, and pdfMark (or maybe pdfmark without the intercap—the naming isn't consistent), a system created by Adobe and described this way:

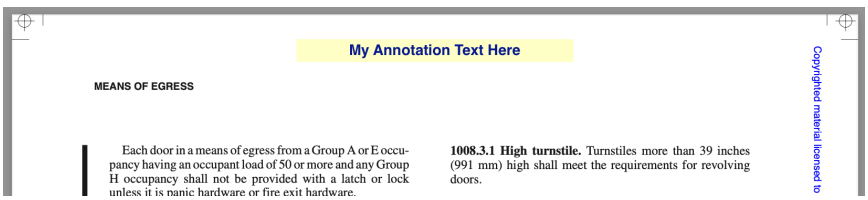
> The pdfmark operator is a PostScript-language extension that describes features that are present in PDF, but not in standard PostScript.

Basically, pdfMark was a way for Adobe's Distiller application to add PDF-specific bits (like annotations) to PostScript files as it was converting them to PDF. It came out back in the 90s, when PostScript was well established, but PDF was still in its infancy. We can use Ghostscript in place of Distiller.

Let's say we have a PDF that consists of several letter-sized pages, and we want to add some text centered in the top margin. If our original looks like this,



we want the annotated version to look like this,



where the annotation appears on the first page only. Here's what we do:

First, create a text file (we'll call it `pdfmark.txt`, but the name can be anything) with the following contents:

Site search

via DuckDuckGo
Go!

Meta

drdrang at leancrew
Blog archive
RSS feed
JSON feed
Mastodon
GitHub repositories

Recent posts

Better medication tracking
Doppler Petty
Converting lists
The Prime Directive
Rolling and pulling
ChatGPT and reliability
ChatGPT and beam bending
Parabolic mirrors made simple(r)
ImageOptim
Feet, inches, and averaging
Better MathJax equations
Concrete and pi

Credits

MathJax

```
1 [
2 /Subtype /FreeText
3 /SrcPg 1
4 /Rect [206 758 406 774]
5 /Color [1 1 .75]
6 /DA (/HeBo 14 Tf 0 0 .5 rg)
7 /Contents (My Annotation Text Here)
8 /ANN pdfmark
```

Without line numbers

This file can be saved anywhere, but for convenience we'll assume it's in the same folder as the original PDF, which I will cleverly name `original.pdf`.

Now we run this Ghostscript command,

```
gs -dBATCH -dNOPAUSE -dQUIET -sDEVICE=pdfwrite -sOutputFile=ann
```

and we end up a new PDF, `annotated.pdf`, with the annotation shown above.

You can probably figure out what each line of `pdfmark.txt` does, but let's run through it anyway.

The opening bracket on Line 1 is the necessary start of every pdfMark command. If you go looking for the matching closing bracket, you won't find one. If you want to know why there's no closing bracket, you'll have to ask Adobe. Seems like really dumb syntax to me.

Line 2 declares this mark to be of the FreeText subtype. There are over a dozen subtypes you can use; see page 16 of the manual.

Line 3 tells the annotation to appear on the first page of the output document. As far as I can tell, there's no convenient way to extend this command to multiple pages. `/SrcPg` can only be followed by a single integer argument, so if you want the same thing on several pages, your `pdfmark.txt` file will have to have this command repeated for each page.

Lines 4 and 5 define the bounding box for the annotation and set its background color. PostScript and PDF coordinates are in points (1/72 inch) with the origin at the lower left corner of the page. Unlike a lot of graphics formats, but like most graphs you see in math class, the y-coordinate increases as you go *up*. A letter-sized page is 612 points wide and 792 points tall, so the bounding box in Line 4 is 200 points wide, centered left/right, and its top edge is ¼ inch down from the top of the page. Colors are defined by an red-green-blue triplet of numbers that run from 0 to 1. Black is `0 0 0` and white is `1 1 1`. White is the default, so if you leave out Line 5, it's equivalent to `/Color [1 1 1]`

Line 6 defines the font used in the annotation. `/DA` means *default appearance*, and the rest of the line tells the text to appear in 14-point Helvetica Bold with a dark blue color.

Line 7 defines the text of the annotation between the parentheses.

Finally, Line 8 identifies the type of pdfmark as an annotation.

(There's a nice document with other examples of pdfMark commands and another Adobe reference manual.)

So how do we automate this? Fundamentally, we create a temporary file for the pdfMark commands, run the Ghostscript command, and then delete the temporary file. Here's a quickly written script, `annotatePDF`:

```
1  #!/usr/bin/env python
2
3  import os
4  import subprocess
5  import sys
6  import tempfile
7
8  # Set the parameters
9  annText = sys.argv[1]
10 originalPDF = sys.argv[2]
11 annotatedPDF = sys.argv[3]
12
13 # Build the pdfMark command
14 pdfMarkCommand = f"""[
15 /Subtype /FreeText
16 /SrcPg 1
17 /Rect [156 758 456 774]
18 /DA (/HeBo 14 Tf)
19 /Contents ({annText})
20 /ANN pdfmark
21 """
22
23 # Create a temporary file for the pdfMark commands and write
24 fh, fpath = tempfile.mkstemp()
25 with open(fpath, 'w') as f:
26   f.write(pdfMarkCommand)
27
28 # Run the Ghostscript command to make the annotated file
29 subprocess.run(['gs', '-dBATCH', '-dNOPAUSE', '-dQUIET', '-s
30
31 # Delete the pdfMark command file
32 os.remove(fpath)
```

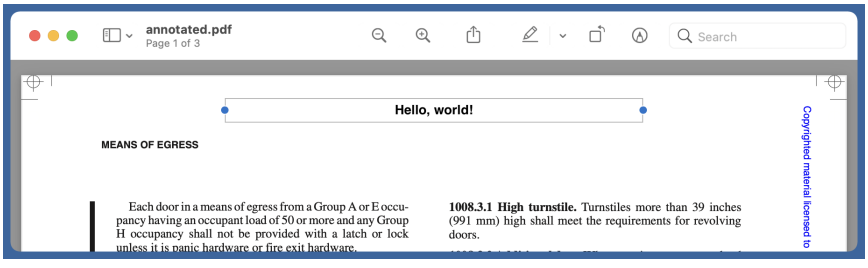
Without line numbers

This is not a great script. No error handling, no options, and no way to automate the naming of the output file. But it works.

```
annotatePDF 'Hello, world!' original.pdf annotated.pdf
```

As you can see from Lines 14—21, I don't really want a yellow box with dark blue text. That was just to show some of pdfMark's features.

By the way, the annotations you add through pdfMark are just like annotations you add in Preview or PDFpen. They can be selected, moved around, and edited. Here's what the output file looks like in Preview after clicking on the added text.



Previous post

Next post