Compressing PDF files

John C. Nash

29 November 2021

Why compress PDF files?

As documented in https://en.wikipedia.org/wiki/PDF, PDF or Portable Document Format files are a way to present documents independent of operating system or application software. In this article, I will be concerned with documents that are assumed to be "standard", though I have not verified this assumption in any of the experiments described.

Many applications and operating system features permit output as PDF files. However, the choices made in producing these files often lead to large file-sizes. Since documents are frequently shared over email and other communications methods, it is helpful if the file sizes are as small as possible. Hence a motivation for compression. On the other hand, if such compression degrades images, particularly images of historical documents, then caution in applying the compression is in order.

Many suggestions (see next section) for compressing PDF files are online services. If we have scanned a personal document, for example a passport or driver's licence, it seems highly dangerous to upload the scan to what may easily be a site that is either criminal or compromised by criminals. We want a program that is run locally.

Test files

I used the following four test files:

- F.pdf – this file represents the scans of some letters of a family member over a period of several years. The 300dpi scanned images were converted to PDF and combined into a single document using ImageMagick `convert` and Linux pdf tools. Size=66041050 bytes.
- HQR.pdf – this file is a cups-pdf "print" of an amazon.ca description of a 6V replacement battery. Size=963515 bytes.
- nlpor.pdf – this is the PDF output of LaTEX of my 2014 book **Nonlinear Parameter Optimization using R Tools** which has 304 pages. There are limited graphics (and no cover image), so the body of the PDF is likely all text and font names, with the fonts NOT included. Size=304086 bytes.
- W.pdf - this is the Wiley "Instructions for Authors of Wires reviews" document and macro template exported to PDF via LibreOffice. Size=100732 bytes.

Online offerings for PDF compression

In my few experiments, "free" online sites limit use either by size of file or number of files that can be compressed. In fact, most seem to be a "come on" to get users to sign up and pay money for services that open-source tools will carry out.

The following are the experiments I did carry out.

- https://pdfstuff.com/compress-pdf/# – fails for F.pdf. It shows an upload progress bar, but stalls at around 58 %

- https://www.freepdfconvert.com/compress-pdf using “Better compression” option for F.pdf gives a result F-freepdf.pdf of size F-freepdf.pdf. The on-screen appearance of the reduced file seems very good, but this is NOT a free site. Attempts to upload another file offer a sign-up or a wait of an hour. (They appear to be using IP number, as changing browser still gave the delay.)
- adobe.com has a similar site. Even removing cookies for adobe forced the browser to a sign-in or sign-up page on a second try. HQR.pdf was reduced to 223260 bytes when it was uploaded on the single try I made. Moreover, the resulting screen presentation was fuzzy.

Further attempts with online tools were abandoned, as it seems almost certain that the sites are data-mining.

Suggested approaches

Searches on the Internet yield many suggestions for proprietary programs, including Adobe Acrobat (??Registered symbol??) (the program rather than the online service). In this article, I am primarily interested in solutions that are

- locally run
- open source
- work in Linux.

I would be interested to know if there are parallel solutions for Windows, MacOS or other platforms, and will gladly collaborate, giving authorship, to detail such solutions and extend this article.

Using the search string “linux PDF compression” yields a number of hits, some of which are inappropriate. However, the following were helpful.

https://itsfoss.com/compress-pdf-linux/

Like many other sites, this recommends using Ghostscript (/url{https://www.ghostscript.com/}) as a tool to compress PDF files.

```
gs -sDEVICE=pdfwrite /
   -dCompatibilityLevel=1.4 /
   -dPDFSETTINGS=/prepress /
   -dNOPAUSE /
   -dQUIET /
   -dBATCH /
   -sOutputFile=compressed_PDF_file.pdf /
   input_PDF_file.pdf
```

In the above command, the user must add the correct path of the input and output PDF files. Moreover, the key argument in this command to control the compression level is `dPDFSETTINGS`. The options for this control are

- /prepress – this is claimed to be the default setting (indeed you can enter /default) and gives a higher quality output (300 dpi) but bigger file size. Other sites suggest /printer, which seems to give similar results (see https://pdf.wondershare.com/pdf-knowledge/compress-pdf-linux.html).
- /ebook Medium quality output (150 dpi) with moderate output file size
- /screen Lowest quality (72 dpi) but highest compression

The site also offers a GUI (programmed in Python) tool from a related Github project https://github.com/itsfoss/compress-pdf, though I found this to be unfinished in source code form, with TODO as the only content of the Read.Me file, and no instructions for building the package. On the other hand, there is a link to an appimage (https://appimage.org/) program **compress-pdf-v0.1-x86_64.AppImage** which works quite well and is easy to use. However, I found it easier for my experiments to use the command-line form, for which I saved pre-coded variants as `spdf-default`, `spdf-printer`, `spdf-ebook`, and `spdf-screen` in my home directory `bin` folder.

https://www.shellhacks.com/linux-compress-pdf-reduce-pdf-size/

This web-page recommends using the **ps2pdf** program available in most Linux distribution repositories. In Linux Mint, it is included in either the `texliv-latex-recommended` or `texlive-latex-extra` packages. The command to carry out compression is simply

```
ps2pdf F.pdf F-ps2pdf.pdf
```

That is, compression is somehow automatic. As we shall see, the results of this are only moderately successful.

My image recoding approach - pdfreducer.sh

My own script for reducing the size of PDF files arose because I have been combining scanned document pages into single PDF files. The page scans are generally saved as JPEG files from an IPEVO Ziggy-Cam document camera. I have a script, **jpgsindir2pdf**, that converts all the files with extension `jpg` in a specified directory to a single PDF file, then opens the program **pdfarranger** (formerly **pdfshuffler**) to allow rotation and some other edits.

The Ziggy-cam has a top resolution of 2592 x 1944 pixels. This is generally equivalent to 300 dpi scan resolution on a flatbed scanner. By experience, this is more than adequate for capturing detail in any documents in which I am interested. I capture in colour at default colour depth of the camera or scanner. This results in quite large files, typically 2 - 5 megabytes even as a JPEG. Moreover, it captures **images**, even if the original is plain text.

The central idea of `pdfreducer.sh` (see the Appendix for a listing) is to use ImageMagick `convert` to change the image density of each page. If we assume 300 dpi in the raw files, and set `DEN=200` in the script (this is the default), we may expect a reduction of the pdf to $(2/3)^2$ or approximately .44 of the original.

It is important to note, however, that this process will be worse than doing nothing for some files.

Comparison of results

Table 1: File sizes observed

	FILE:	F	HQR	nlpor	W
Method					
original size		66041050	963515	304086	100732
freepdfconvert.com		2377592			
adobe.com			223260		
ps2pdf		66043576	711744	228907	77832
spdf-default		66043535	711703	228869	77792
spdf-printer		66047088	941867	238209	96583
spdf-ebook		3004947	357083	225029	46420
spdf-screen		1008612	211302	225408	33173
pdfreducer		13566502	1960712	19644715	301853

We can compute the ratio of "compressed" to original very easily. This is presented in Table 2 as a percentage to 1 decimal. We note that `pdfreducer` does appallingly poorly on some files. It appears that `ps2pdf` is essentially the same method as GhostScript default method `spdf-default`. Both `spdf-ebook` and `spdf-screen` do a very decent job of PDF reduction, but we must be careful.

Figure 1 is a very crude attempt using LibreOffice Draw to capture parts of the original, `spdf-ebook` and `spdf-screen` PDF files as they are seen on the screen. I make no grand claims for the reliability of this approach, but it does indicate that the "screen" option is likely too aggressively reduced, since there is a fuzziness in the resulting output that is easily visible. The "ebook" option does appear to be more or less satisfactory for most of the applications I have, though I would NOT rely on it for saving archival material.

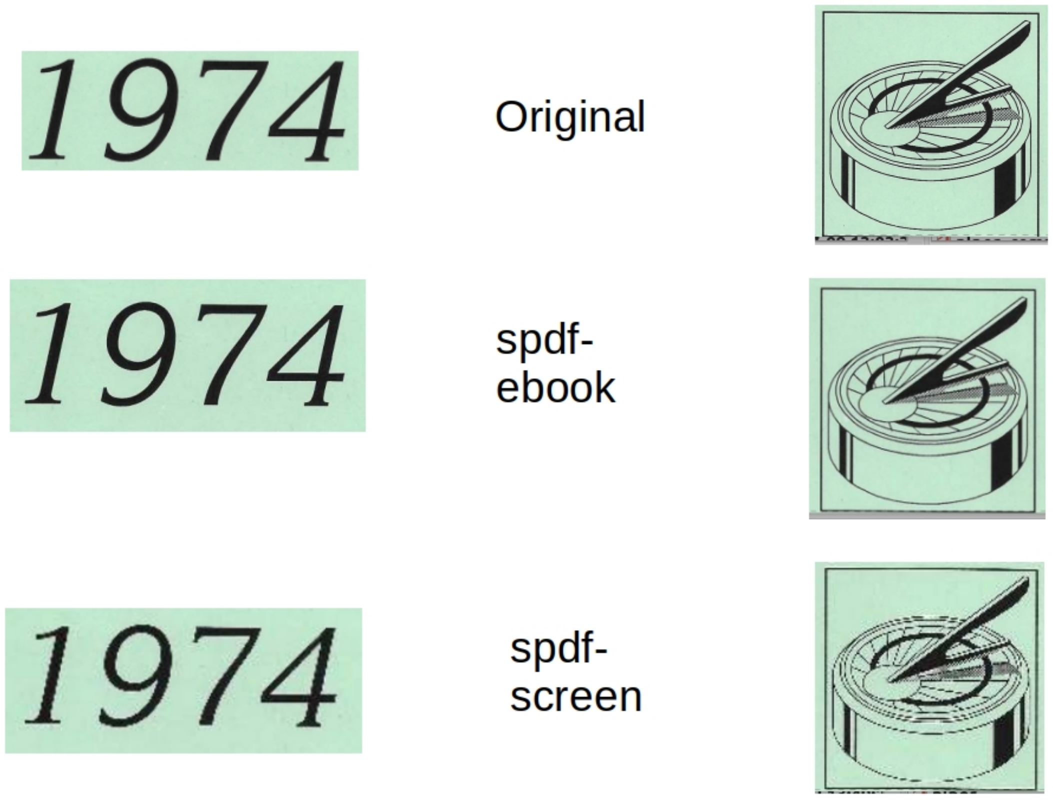

Figure 1: Crude output quality comparison via screen capture

Table 2: Compression ratios = 100 * newsize / original-size

FILE:	F	HQR	nlpor	W
Method				
freepdfconvert.com	3.6	0.0	0.0	0.0
adobe.com	0.0	23.2	0.0	0.0
ps2pdf	100.0	73.9	75.3	77.3
spdf-default	100.0	73.9	75.3	77.2
spdf-printer	100.0	97.8	78.3	95.9
spdf-ebook	4.6	37.1	74.0	46.1
spdf-screen	1.5	21.9	74.1	32.9
pdfreducer	20.5	203.5	6460.2	299.7

My tentative conclusion is that I will likely use `spdf-ebook` for reducing the size of PDF files when I need to email them or put them on web-sites that are non-archival. I will likely use my own `pdfreducer` script when I know that the documents to be reduced have come from scans or document camera page images. The output from `pdfreducer` in such cases is a significant reduction in storage without appreciable compromise in image quality.

Extension and use

If `spe` is used as a standard tool, it can be used to reduce storage space for PDF files in archived collections. This was tested with a collection called `jkeep` by

- making a list of all files in the directory that have the extension (in either upper, lower or mixed case) `.pdf`.
- applying `spe` to these.

This process reduced the storage from 9835680 to 7809820 bytes. However, a few (in fact, 38) applications of `spe` generated errors and the output files were unsatisfactory. There were also cases where it did not compress files. Of 4058 PDF files, 3325 were reduced in size (though possibly in some cases with errors), while 733 were left unchanged.

While it should be possible to detect the errors when Ghostscript (i.e., `gs`) is run, my experiments so far have shown that the `gs` command does not yield a non-zero return code when GhostScript emits error messages to the stderr or stdout channels. Without the return code, we must examine this output for character strings that indicate an error. I have, however, found cases (e.g., ScanComfortsuites160103.pdf) for which GhostScript shows an error, but the shrunk pdf is apparently satisfactory.

These ideas have been incorporated into the single file PDF shrink script `spe`, which is then called by the directory tree processor `dirtreespe`. However, my opinion is that this automated compression of a block of files should not be entirely trusted as yet (November 29, 2021).

Appendix: Scripts mentioned

jpgsindir2pdf

```
#!/bin/bash
# jpgsindir2pdf -- convert jpgs to pdf and name for directory

echo "jpgsindir2pdf"

SAVEIFS=$IFS
IFS=$(echo -en "\n\b")
```

```
CDIR=$PWD
echo "CDIR=$CDIR"
#read tmp
echo "parameter=$1"
NAME=$(basename "$1") # base of file name
echo "NAME=$NAME"
#read tmp
cd $NAME
pwd
echo "have we changed directory?"
#read tmp

FILES=*.jpg
for f in $FILES
do
  TT=`basename -s .jpg $f`
  convert $f $TT.pdf
  echo "$TT.pdf"
done

pdftk *.pdf cat output ../$NAME.pdf
echo "Created $NAME.pdf"
cd ..

pwd
ls

echo "now arrange"
#read tmp

pdfarranger $NAME.pdf

echo "delete intermediate pdf files"
read tmp
rm $NAME/*.pdf

IFS=$SAVEIFS
echo "done!"
```

pdfreducer

```
#!/bin/bash
# pdfreducer.sh
DEN=200
## standardizes density at $DEN
## make current working directory == must change name
##   or will overlap
WD=wd`date +%s`
echo "working dir=$WD"
# read tmp
## temporary working directory
## Be nice to show a sign that process is working
mkdir $WD
## save file to working directory
```

```
cp $1 $WD/
cd $WD/
## then split apart
pdftk $1 burst
## remove working file
rm $(basename "$1" .pdf).pdf
## convert to jpg with appropriate density
for fn in ./pg*.pdf; do
    convert -density $DEN $fn $(basename "$fn" .pdf).jpg
done
rm pg*.pdf
convert pg*.jpg new.pdf
# ls -al
# read tmp
echo "remove temporary pdfs and working directory"
read tmp
cd ..
cp $WD/new.pdf ./
mv new.pdf $(basename "$1" .pdf).red.pdf
rm -r $WD
```

shrinkpdf.sh

This version was based on merged suggestions found on the Web.

```
#!/bin/sh
# JN version of spdf-ebook == spe  2021-11-28
# OIFS=$IFS # doesn't seem to work in Mint
# echo "IFS=$IFS"
# change IFS to only use newline
# IFS='
# '
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/ebook -dNOPAUSE -dQUIET -dBATCH -sOutputFil

grep "**** Error" tmplog.txt > tttt
size=$(wc -c <"tttt")
if [ $size != 0 ]; then
   echo "FAIL"
   exit 1
else
   echo "NO ERROR"
fi
# get sizes
A=$((`du --bytes out.pdf | cut -f1`))
echo "New file is $A bytes"
B=$((`du --bytes $1 | cut -f1`))
# note the brackets to convert string to number
echo "Original is $B bytes"
if [ $B -gt $A ];
then
    echo "Success! -- original in $1.orig"
    mv $1 $1.orig
    mv out.pdf $1
else
    echo "Unchanged!"
    rm out.pdf
fi
# IFS=$OIFS
```

spdf-default

```
#!/bin/sh
# JN version of spdf-default
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/default /
  -dNOPAUSE -dQUIET -dBATCH -sOutputFile=out.pdf $1
mv out.pdf $1.spdf-d
```

spe (formerly spdf-ebook)

This script has been upgraded as the one most likely to be used. It has also been renamed `spe` for ease of typing.

```
#!/bin/sh
# JN version of spdf-ebook == spe
# OIFS=$IFS # doesn't seem to work in Mint
# echo "IFS=$IFS"
# change IFS to only use newline
# IFS='
# '
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/ebook -dNOPAUSE -dQUIET -dBATCH -sOutputFi

# get sizes
A=$((`du --bytes out.pdf | cut -f1`))
echo "New file is $A bytes"
B=$((`du --bytes $1 | cut -f1`))
# note the brackets to convert string to number
echo "Original is $B bytes"
if [ $B -gt $A ];
then
    echo "Success! -- original in $1.orig"
    mv $1 $1.orig
    mv out.pdf $1
else
    echo "Unchanged!"
    rm out.pdf
fi
# IFS=$OIFS
```

spdf-prepress

```
#!/bin/sh
# JN version of spdf-prepress
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/prepress /
   -dNOPAUSE -dQUIET -dBATCH -sOutputFile=out.pdf $1
mv out.pdf $1.spdf-pre
```

spdf-printer

```
#!/bin/sh
# JN version of spdf-printer
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/printer /
```

```
   -dNOPAUSE -dQUIET -dBATCH -sOutputFile=out.pdf $1
mv out.pdf $1.spdf-pri
```

spdf-screen

```
#!/bin/sh
# JN version of spdf-screen
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/screen /
   -dNOPAUSE -dQUIET -dBATCH -sOutputFile=out.pdf $1
mv out.pdf $1.spdf-s
```

dirtreespe – shrink pdfs in a directory tree

```
#!/bin/sh
# JN version of spdf-ebook on a tree == dirtreespe
## $1 gives the top tree dir
TT=`tstamp.sh`
# echo $TT
# echo "${TT} spe error file"
FF="${TT}speErrors.txt"
echo $FF
echo "$TT spe error file" >$FF

szold=`du -s $1 | cut -f1`
echo "globbing $1  size=$szold"
find $1 -iname "*.pdf" > glob.txt
i=0
while read f;
do
  i=$((i+1))
  # note how incrementing is done
  echo "$i : $f"
# reduce size
  # `spe $f`
  # note the quotes to avoid spaces
  # also cannot seem to call my own script, but cp works
  cp "$f" /tmp/w.pdf
  spe /tmp/w.pdf
  ret=$?
  echo "return code=$ret"
  if [ "$ret" = '0' ]; then
      echo "Save back $f"
      mv /tmp/w.pdf "$f"
      mv /tmp/w.pdf.orig "$f.orig"
  else
      if [ "$ret" = '1' ]; then
         echo "Error for $f"
         echo "$f" >> $FF
      else
         echo "Unchanged for $f"
      fi
  fi
#  rm "$f.orig"
done <glob.txt
echo "============================="
```

```
mv glob.txt "../${TT}glob.txt"
sznew=`du -s $1 | cut -f1`
echo "Size Before=$szold  After=$sznew"
echo "DONE!"
```