Advanced-PDF-Tricks

This repository (for now) is development home to some hand-crafted PDF files. These PDFs should serve as study material for everybody who wants to learn about this format.

Students of the PDF file formats should be able to use a *text editor* in order to open and change them.

Some of the samples include commentary to give hints how to start experimenting with them. These can mostly be changed by overwriting `%` comment characters with a space character (and maybe comment out another line instead) in order to see how the PDF viewer changes its display.

Most of the PDFs don't have the fonts embedded which they may use (Courier, Times, Helvetica). This may cause that some viewers or Ghostscript complain that they cannot find a substitute font.

Acknowledgements

The initial creation of these files was inspired by a talk held at the TROOPERS15 conference in Heidelberg.

Text editor hints

Make sure that your editor does not change the end-of-line conventions used by the original PDF. Windows uses `[CARRIAGE_RETURN]+[LINE_FEED]` (2 bytes: in HEX `0D 0A`), while Linux and OS X use `[LINE_FEED]` (1 byte only: `0A` as HEX).

If you don't take care of that, the edited PDF may become "corrupted" by simply opening the file and saving it again. The reason would be because the file's internal *table of content* (so-called `xref`-table) would point to wrong byte offsets by the changed numbers of bytes.

VIm users should start their editor with `vim -b` (for binary mode) to make sure the internal byte counting works correctly.

[… to be continued …]

PDF viewer hints

Some viewers reload the PDF file and change their page display of an opened file "on the fly" as soon as they notice a change (triggered by editor saving the file). Amongst these are the venerable `gv`, SumatraPDF (Windows only), MuPDF (all platforms), Zathura (Linux only) and in parts Preview.app (OSX only) too. Acrobat (and Adobe Reader) need to be closed and restarted with the changed file before you can see any edit effect.

Compressing/Uncompressing data blobs with Zlib algorithm

You can use OpenSSL to compress/uncompress data blobs using the Zlib algorithm:

```
openssl zlib -d < $IN > $OUT
```

ZLIB un-/compression is equivalent to PDF's *Flate* de-/encoding.

Note: the `zlib` sub-command (as well as the `-z` option to the `enc` sub-command) is not available if your build of OpenSSL was configured with the default options. Unfortunately these include `--no-zlib` and `--no-zlib-dynamic`. So this trick only works if your OpenSSL was compiled with the `no-` prefix removed from one of those configure options. You can tell by looking for `-DZLIB` in the output of `openssl version -f`.

Compressing/Uncompressing data blobs with Zlib algorithm

The following Python one-liner should achieve the same:

```
python -c "import zlib,sys;print \
        repr(zlib.decompress(sys.stdin.read()))" < $IN
```

Compressing/Uncompressing data blobs with the `zlib-flate` command

There is an additional command line utility shipping with **qpdf**: its name is `zlib-flate`. Here is how to use it:

```
zlib-flate -compress   < $IN  > $OUT     # to compress $IN and generate $OUT
zlib-flate -uncompress < $OUT > $IN      # to uncompress again
```

or

```
cat $IN | zlib-flate -uncompress         # to uncompress $IN
```

Copyright (c) 2015 kurt.pfeifle@mykolab.com

License: Creative Commons "CC-BY-NC-SA" v4.0