hhrutter / **pdfcpu**

A PDF processor written in Go.

#go #golang #golang-library #pdf #processor #pdf-processor #pdf-files

121 commits | **1** branch | **26** releases | **4** contributors | Apache-2.0

Branch: **master** | New pull request | Find file | Clone or download

hhrutter fix stream parsing — Latest commit `5dbcc5d` 6 days ago

_scripts	stamping fixed	8 days ago
ccitt	fix extract for CCITT	24 days ago
cmd/pdfcpu	v0.1.18 fix for Travis	8 days ago
lzw	wip	18 days ago
pkg	fix stream parsing	6 days ago
resources	v0.1.18	8 days ago
tiff	wip	18 days ago
.gitignore	cleanup	9 months ago
.travis.yml	watermarks & Apache-2.0	2 months ago
LICENSE.txt	watermarks & Apache-2.0	2 months ago
README.md	v0.1.18 fix for Travis	8 days ago
coverage.sh	all types refactored other than PDFDict	2 months ago

README.md

pdfcpu: a golang pdf processor

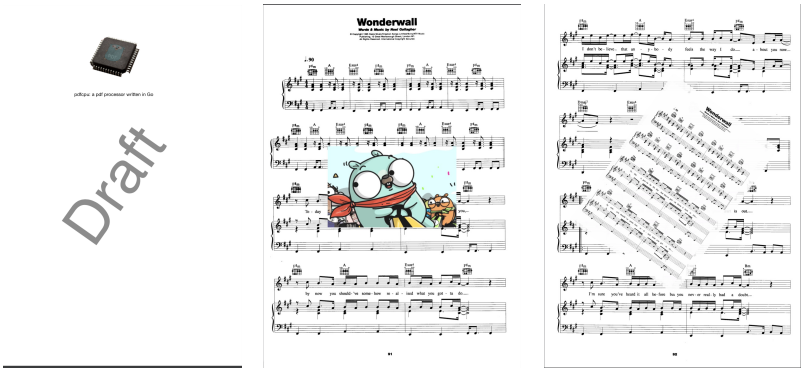
build passing | godoc reference | coverage 87% | go report A+ | license Apache 2

Package pdfcpu is a simple PDF processing library written in Go supporting encryption. It provides both an API and a CLI. Supported are all versions up to PDF 1.7 (ISO-32000).

Status

Version: 0.1.18

- Extended API to support webserver scenarios using Readseeker and Writer.
- Support for watermarking/stamping with a specific page of another PDF file.
- Extended logging into horizontal (Info, Debug, Trace etc.) vs. vertical logging (Read, Validate, Write etc).
- The CLI will produce regular logging if you use -verbose, or -v.
- The CLI will produce verbose logging if you use -vv.
- More tests in `api/process_test.go`
- More examples in `api/example_test.go`
- More scripts under `_scripts/*`
- Fixed #5, #39, #44



Motivation

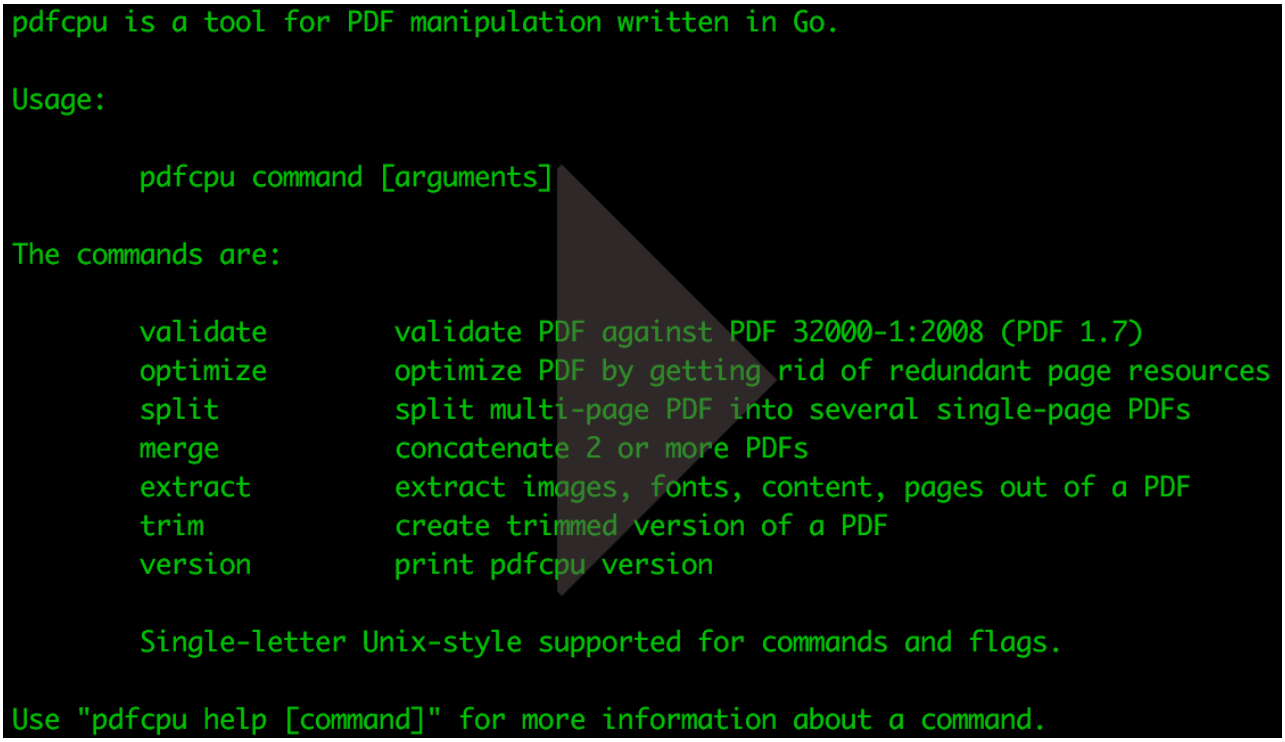
This is an effort to build a PDF processing library from the ground up written in Go with strong support for batch processing via a rich command line. Over time `pdfcpu` aims to support the standard range of PDF processing features and also any interesting use cases that may present themselves along the way.

One example is reducing the size of large PDF files for mass mailings by optimization to the bare minimum. This can be achieved by analyzing a PDF's cross reference table, removing redundant embedded resources like font files or images and by always writing back the file maxing out PDF compression. I also wanted to have my own swiss army knife for PDFs written entirely in Go that allows me to trim, split, stamp and merge PDF content.

Features

- Validate (validates PDF files up to version 7.0)
- Read (builds xref table from PDF file)
- Write (writes xref table to PDF file)
- Optimize (gets rid of redundancies like duplicate fonts, images)
- Split (split a multi page PDF file into single page PDF files)
- Merge (a set of PDF files into one consolidated PDF file)
- Extract Images (extract all embedded images of a PDF file into a given dir)
- Extract Fonts (extract all embedded fonts of a PDF file into a given dir)
- Extract Pages (extract specific pages into a given dir)
- Extract Content (extract the PDF-Source into given dir)
- Extract Metadata (extract XML metadata)
- Trim (generate a custom version of a PDF file)
- Stamp/Watermark selected pages with text, image or PDF page
- Manage (add,remove,list,extract) embedded file attachments
- Encrypt (sets password protection)
- Decrypt (removes password protection)
- Change user/owner password
- Manage (add,list) user access permissions

Demo Screencast (this is an older version with a smaller command set)



Installation

Required build version: go1.9 and up

```
go get github.com/hhrutter/pdfcpu/cmd/...
```

Usage

```
pdfcpu validate [-verbose] [-mode strict|relaxed] [-upw userpw] [-opw ownerpw] inFile
pdfcpu optimize [-verbose] [-stats csvFile] [-upw userpw] [-opw ownerpw] inFile [outFile]
pdfcpu split [-verbose] [-upw userpw] [-opw ownerpw] inFile outDir
pdfcpu merge [-verbose] outFile inFile...
pdfcpu extract [-verbose] -mode image|font|content|page|meta [-pages pageSelection] [-upw userpw] [-opw
ownerpw] inFile outDir
pdfcpu trim [-verbose] -pages pageSelection [-upw userpw] [-opw ownerpw] inFile outFile
pdfcpu stamp [-verbose] -pages pageSelection description inFile [outFile]
pdfcpu watermark [-verbose] -pages pageSelection description inFile [outFile]

pdfcpu attach list [-verbose] [-upw userpw] [-opw ownerpw] inFile
pdfcpu attach add [-verbose] [-upw userpw] [-opw ownerpw] inFile file...
pdfcpu attach remove [-verbose] [-upw userpw] [-opw ownerpw] inFile [file...]
pdfcpu attach extract [-verbose] [-upw userpw] [-opw ownerpw] inFile outDir [file...]

pdfcpu encrypt [-verbose] [-mode rc4|aes] [-key 40|128] [-perm none|all] [-upw userpw] [-opw ownerpw]
inFile [outFile]
pdfcpu decrypt [-verbose] [-upw userpw] [-opw ownerpw] inFile [outFile]
pdfcpu changeupw [-verbose] [-opw ownerpw] inFile upwOld upwNew
pdfcpu changeopw [-verbose] [-upw userpw] inFile opwOld opwNew

pdfcpu perm list [-verbose] [-upw userpw] [-opw ownerpw] inFile
pdfcpu perm add [-verbose] [-perm none|all] [-upw userpw] -opw ownerpw inFile

pdfcpu version
```

Please read the documentation

Contributing

- Please open an issue if you find a bug or want to propose a change.
- Feature requests - always welcome
- Bug fixes - always welcome

- PRs - also welcome, although I can't promise a merge-in right now since `pdfcpu` is stable but still *alpha* and occasionally undergoing heavy changes.

Disclaimer

Usage of `pdfcpu` assumes you know about and respect all copyrights of any PDF content you may be processing. This applies to the PDF files as such, their content and in particular all embedded resources like font files or images. Credit goes to Renee French for creating our beloved Gopher.

License

Apache-2.0

Powered By