pdfslash

Open Close

Apr 14, 2022

CONTENTS:

Crop PDF margins from interactive cui.

CHAPTER

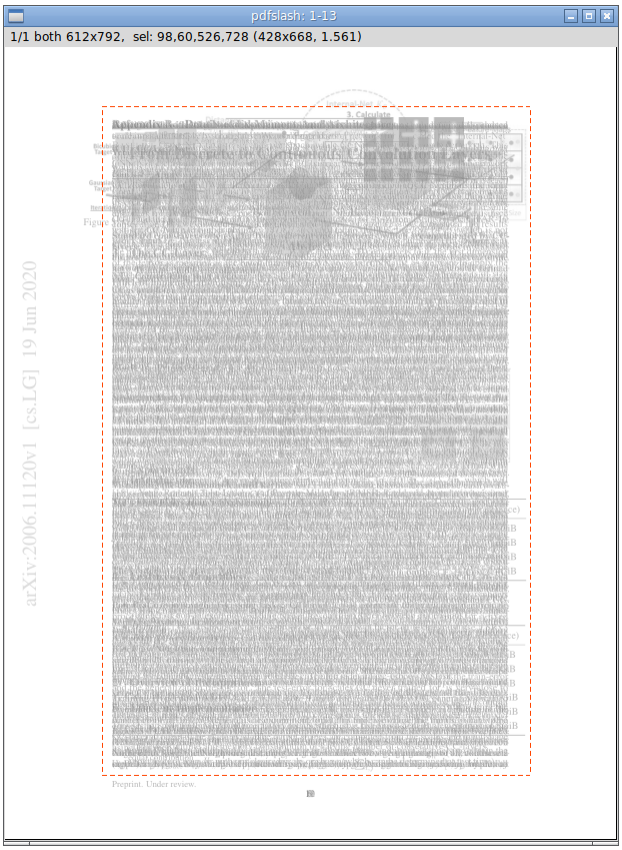
ONE

INTRODUCTION

This Python program helps batch-edit PDF page margins.

It runs an interactive interpreter as an intermediate layer. Various commands, including GUI, are called from there.

Intended to create more readable PDF documents for e-reader devices, for personal reading.



1.1 Technical Overview

- Use PyMuPDF to process PDF.
- Use Numpy to process image pixel data.
- Borrow algorithms from briss, for key functions (page merging and auto margin detection).
- interpreter and GUI are using Python standard libraries (cmd and tkinter).

1.2 Installation

```
$ pip install pdfslash
```

The command installs the program, a single pure Python module. But you also need to install `PyMuPDF` and `Numpy` yourself.

(To use `'info'` interpreter command, `PyMuPDF v1.18.7` or later is required).

1.3 Main Workflow

It requires one argument, pdf filename.

```
$ pdfslash some.pdf
```

Upon invocation, users are faced with a commandline prompt.

```
(pdfslash) <-- this
```

Interpreter Commands:

Most commands only take one argument (page numbers), box commands take two or three.

```
(pdfslash) show 1-3,7

    # Show box data for page 1,2,3,7

(pdfslash) crop 6-9 30,30,400,500

    # Append new box (left,top,right,bottom),
    # for page 6,7,8,9.
```

GUI:

To run GUI, use `preview`.

```
(pdfslash) preview 1-100
```

In GUI, Pages are grouped by their source mediabox and cropbox sizes (See Spec `preview` command for details).

Pages of each group are merged into one grayscale image, as in `briss`.

Each image size is equal to their mediabox size.

Only one (initially first) image is shown in GUI window. You can navigate by keys `n` and `p` (next and previous).

Each image has actually three views, `all`, `odds` and `evens`. E.g. if the pages of a group is `2-5,9`, `all` is `2,3,4,5,9`, `odds` is `3,5,9` and `evens` is `2,4`. You can cycle them by keys `v` and `V` (forward and backward).

Source cropboxes are shown in green. You can toggle visibility (show and hide) by key `s` (source cropbox).

But when cropbox is equal to mediabox (in most cases), you can not see it anyway.

If there are previously created boxes (say, previous boxes), they are also shown. Boxes in all pages (in this group and view) are in blue, and boxes just in some pages are in a bit lighter blue.

Crop:

In any of group and view, you can create a new box (future cropbox) with mouse. Click (top-left), drag and release (bottom-right). The created box is shown with dotted lines.

If you want to delete the box, just create a new one, or do a short click (too small rectangles are removed).

Adjust the box with cursor keys. `Left`, `Right`, `Up` and `Down` keys move top-left point by one pixel. With `Shift` (`Shift + Left` etc.), the keys move bottom-right point. With `Control`, the keys move the box as a whole.

To register the box, press `Enter (Return)` or `Shift + Enter`. Until then, the box exists only as GUI graphic. The dotted lines of box will change to solid lines.

When just pressing `Enter`, the new one is appended. The previous boxes are preserved.

With `Shift + Enter`, the new one replaces all the previous boxes. That is, if any, they are removed.

Edit:

You can cycle 'active' box. Initially the active box is `sel`, a virtual box (one with dotted lines created by mouse click, or non-existent).

But Pressing `a`, the active box cycles to one of previous boxes in order, if any. The color changes to orange. In this state, you can edit previous boxes with cursor and `Enter` keys (No `Shift + Enter` key in this case).

With `u` and `r`, the program performs undo and redo. They are only for box registrations, other states are initialized.

To exit GUI, press `q`.

Save:

To actually create a new PDF file, use `write`.

```
(pdfslash) write 2-48
```

It creates a new file with `'.slashed'` appended to the filename. (E.g. `some.pdf` to `some.slashed.pdf`).

CHAPTER

TWO

SPEC

2.1 Box

In this program, Box values are always expressed as (left, top, right, bottom), in the coordinates in which rotation is already applied, y-descendant, `MediaBox`'s left-top moved to (0, 0), floats clipped to integers.

User created boxes must also be integers.

In addition, they must be unique in a page. Duplicate boxes (the same boxes in a page) are not possible.

2.2 Config

If there is a environment variable `'PDFSLASH_DIR'` and it is a directory, or `'$XDG_CONFIG_HOME/pdfslash'` or `'~/.config/pdfslash'` is a directory, the program registers it as user-directory.

If there is a user-directory, and the system can use Python standard library readline, the program uses a file `'.history'` (automatically created) for the readline's history file (And `.python_history` for `Python` command).

If there is a file `'pdfslash.ini'` in the directory, the program reads it and update the configuration.

The defaults are:

```
[main]

# The ratio of computer display pixel, to PDF pixel.
device_pixel_ratio = 1.0

# Gui window position to the display margin (x and y).
# 0.0, 0.0: top-left aligned
# 0.5, 0.5: center
# 1.0, 1.0: bottom-right aligned
winpos = 0.5, 0.5

# Max pages to sample, to create a merge image (one group) in GUI.
# So when running 'preview 1-600',
# the program is acutually showing
# only this number of arbitrarily selected pages.
# '15' is briss' default.
max_merge_pages = 15

# Merge method to use in page-image-merging,
```

(continues on next page)

(continued from previous page)

```
# either 'briss' (default) or 'simple'.
# 'simple' is a bit faster,
# and *may* work in some cases where 'briss' doesn't.
merge = briss
```

2.3 Commandline

-h, --help
: show this help message and exit

PDFFILE
: PDF filename to process

--command COMMAND, **-c** COMMAND
: run initial commands before showing prompt (split multiple commands with ';').

--cmdfile CMDFILE, **-f** CMDFILE
: run initial commands before showing prompt (reading from a file, one command a line).

--nocheck, -n
: do not perform initial commands verification. Otherwise the program aborts when a line starts with '# hash: ', and the value is different from input PDF file. This is for 'export' command.

--_time, -_t
: [DEBUG] print time for some processes

--_save, -_s
: [DEBUG] save merged image used for GUI in current directory

2.4 Interpreter

- Token separator is space, so any command or argument must not have spaces.
- When the command string starts with `'#'`, it is ignored.
- When the command string starts with Python regex `'\[[a-z]+\] '`, the matched part is stripped.

  (e.g. `'[gui] crop 1 10,10,400,500'` -> `'crop 1 10,10,400,500'`).
- Page number syntax is as follows.

```
1-5         1 to 5 inclusive (1,2,3,4,5)
3-          3 to last page
-10         1 to 10

1^5         every other page in 1 to 5 inclusive (1,3,5)
2^6         every other page in 2 to 6 inclusive (2,4,6)
            two operands must be both odds or both evens.

:           all pages

~           the same pages as the previous command
```

- Box syntax is as follows.

```
10,20,30,40     left,top,right,bottom
                (they must be integers, no dots).

For 'modify' command, the following special syntax can be used.

For box1 (box to modify):

@               Specify box with order.
                E.g. '@1' means first boxes for each page.

For box2 (box to modify *to*):

+-              Apply increment or decrement
                to the chosen boxes (by box1) of each page.

                E.g. when box is '-3,-3,+3,+3':

                    20,20,400,400  ->  17,17,403,403
                    30,30,600,600  ->  27,27,603,603

min, max        min or max numbers
                of the chosen boxes (by box1) of each page.

                E.g. min,min,max,+0
                (select the broadest rectangle
                 for left, top and right,
                 but do not change the bottoms.)
```

2.4.1 Commands

- commands are case sensitive (e.g. `Set` and `Python` start with capital letters).
- When commands take *optional* page numbers and they are omitted, *selected* pages are used.
- Admittedly `select`, `unselect`, `fix` and `unfix` tend to get very confusing.

  But normally you don't have to think about them, until when you need them.
- Interpreter and GUI are using the same undo and redo stack data.

  So in interpreter, you can go all back to the initial state, through any changes done in GUI. But in GUI, undo is bound to the GUI invocation, you can't go back past the changes done in the current GUI.

select

Take one argument, page numbers.

`select` page numbers.

Operations are done to only selected pages. Initially all pages are selected.

Use when you don't want to repeat very complex page numbers.

```
unselect :                  # unselect all pages
select 2-8                  # select pages 2-8
```

(continues on next page)

(continued from previous page)

```
crop 1-10 100,100,400,400   # crop pages 2-8
write                       # write pages 2-8
```

unselect

Take one argument, page numbers.

`unselect` page numbers.

See `select`.

fix

Take one argument, page numbers.

`fix` page numbers.

Box operations are not done to fixed pages. Initially all pages are unfixed.

Use when you want to make some pages 'done'.

```
crop 2,3 150,150,450,450    # crop pages 2,3
fix 2,3                     # fix pages 2,3
crop 2-6 100,100,400,400    # crop pages 4,5,6
write 2-10                  # write pages 2-10
```

unfix

Take one argument, page numbers.

`unfix` page numbers.

See `fix`.

append

Take two argument, page numbers and box.

Append box.

(Add box to specified pages, keeping previously added boxes.)

overwrite

Take two argument, page numbers and box.

Replace box.

(Add box to specified pages, removing previously added boxes.)

modify

Take three argument, page numbers, box1 and box2.

Modify box.

(For each page, change pre-existent box (box1) to new box (box2). If box1 doesn't exist in any page, it is Error).

discard

Take two argument, page numbers and box.

Delete box.

(Find the box in each specified page, and remove them. If the box doesn't exist in any page, it is Error).

clear

Take one argument, page numbers.

Clear boxes.

(Delete all added boxes in specified pages. that is, they will revert to the original source cropboxes).

auto

Take one argument, page numbers (optional).

Auto detect page margins and apply (overwrite) them. All previously added boxes are removed.

If the number of previous boxes is one, the detection is done against this box, else (the number is zero or two or more), the detection is done against source cropbox.

preview

Take one argument, page numbers (optional).

Run tkinter GUI.

Options (optional):

-m, --mediabox: Group pages by source mediabox

-c, --cropbox: Group pages first by source mediabox, and then by source cropbox (for each mediabox group). This is the default.

-s, --single: Group each page in each group, to navigate pages one by one.

-_q, --_quit: Create GUI window and immediately quit (for test).

write

Take one argument, page numbers (optional).

Create new PDF file with specified (or *selected*) pages.

It uses PyMuPDF's `fitz.Document.save` method, with the same arguments as `fitz.Document.ez_save`, except 'garbage=2' (instead of '3').

Options (optional):

-m, --more: Shortcut for `-a{'garbage':3}`. For shorter PDF, it seems OK. May make file size smaller, but it tends to get very slower.

-a, --args: Update the default arguments. The string after, say, `-a` must be valid Python code, evaluating to a dictionary, with no spaces.

show

Take one argument, page numbers (optional).

Show current boxes for specified pages.

If selected or fixed, pages are shown with headers `'s'` and `'f'` respectively.

info

Take one argument, page numbers (optional).

[PyMuPDF v1.18.7 or later is required]

Show some PDF information for *specified pages*.

- `Page Count` and `PageLabels`
- `MediaBox`, `CropBox`, `BleedBox`, `TrimBox`, `ArtBox`, `Rotate` and `UserUnit`

For boxes, the (almost) same values from the previous boxes are omitted.

`PageLabels` and `UserUnit` are omitted if they are not defined.

The values are as when PDF file was first loaded. User crop commands don't update them.

Options (optional):

`-p`, `--pdf`: print raw PDF string values as is. In this case, page attribute inheritances are not followed (`MediaBox`, `CropBox` and `Rotate`).

undo

Take no argument.

Undo box operations.

redo

Take no argument.

Redo box operations.

Set

Take zero or two arguments, config option name and option value.

With no argument, show current config options.

With two arguments, set config options

```
Set winpos 0,0
```

Python

Take no argument.

Run Python interpreter, with two variables exposed: `doc` and `pages` (current `Document` and `Document.pages` object).

You are supposed to know the source code.

For now, you can use it only for reading (not writing), otherwise, it will terribly break undo and redo.

(But if you are careful, not using undo and redo, then you *may* be able to save PDF file successfully).

To exit *this* Python interpreter, run `exit()` or send `EOF`.

export

Take no argument.

Print all box edit history in chronological order.

Conceptually, if they are supplied as input again, the program should 'replay' the same edits.

```
(pdfslash) export | cat > log.txt
(pdfslash) exit
$ pdfslash -f log.txt some.pdf
```

It also prints file hash (crc32 to be exact) as comment, and 'replay' will fail if the hash is different from the current input PDF file ('some.pdf' in the example above). You can use commandline option `--nocheck` in this case.

free

Take no argument.

Free all image cache in the program. Use when the program is grabbing too much memory.

(The program caches almost all GUI image and intermediate numpy arrays).

exit

Take no argument.

Exit the program.

crop

Alias for `append`.

quit

Alias for `exit`.

(EOF)

Alias for `exit`. Send actual `EOF`.

'|' (pipe)

If any command has a string `'|'`, the output of the command is passed to the shell.

Intended for a few basic things. E.g.:

```
show 1-100 | grep 155

show 1-100 | cat > log.txt
```

(Currently, in the shell command string after `'|'`, only `'>'`, `'>>'` and `'|'` are considered as shell special tokens. All other special characters are quoted, so they may not work as expected).

2.5 GUI

2.5.1 Info

title bar and label show some information.

title bar example:

```
pdfslash: 1-13,21 (110%) [copy]
```

`1-13,21`: current page numbers (in current group and current view).

`(110%)`: current image zoom (when 100%, it is omitted).

`[copy]`: string `copy`, shown only when copy is pending (after key `c`).

label example:

```
1/3 both 595x841, sel: 100,100,400,500 (300x400, 1.333)
```

`1/3`: current group number (`1`) and the number of groups (`3`).

`both`: current view (`both`, `odds`, or `evens`).

`595x842`: current source mediabox size (GUI canvas size). `left` and `top` are always zeros (`0,0,595,841`).

`sel`: active box (either string `'sel'` or `'box'`).

`100,100,400,500`: active box coordinates.

`300x400`: active box size

`1.333`: ratio of height / width of active box.

2.5.2 Keyboard

```
# <Arrow> means Left, Right, Up or Down keys

mouse:
    left click:     start selection (top-left)
    drag:           expand selection
    release:        end selection (bottom-right)

keys:
    H:              print this help in terminal
    q:              quit

    <Arrow>:        move top-left point
    Shift+<Arrow>:  move bottom-right point
    Control+<Arrow>:move rectangle
    h, j, k, l:     move rectangle (Left, Donw, Up, Right)

    Return:         crop by present selection (append)
    Shift+Return:   crop by present selection (replace)

    n:              next image group
    p:              previous image group
    v:              cycle view (both, odds or evens)
    V:              cycle view (reverse direction)
    s:              toggle souce cropbox visibility

    a:              cycle active rectangle
    d:              delete active rectangle
    c:              copy active rectangle
    z:              zoom in
    Z:              zoom out
    u:              undo (box operations)
    r:              redo (box operations)

(when copy is pending):
    left click:     paste copied rectangle
    x:              paste copied rectangle (the same coords)
```

CHAPTER

THREE

NOTE

3.1 Design

This is the program to manually crop PDF margins, from visual information of merged (superinterposed) images, in GUI.

So, while the program has many interpreter commands, they are rather complementary. Especially,

`auto`: This is actually the first part of the code I've written, but I don't see any good place for it, in general workflow.

box edit commands (`append`, `overwrite`, `modify`, `discard`, `clear`): They are there, mainly for consistency with GUI commands.

They are sometimes useful, but I don't plan to make them particularly easier to use.

The main purpose I see is to be able to 'replay' `export` command output.

Thanks to `MuPDF` and `numpy`, This Python program is quite faster than `briss` (Java).

I'd like to know if there are any PDF-crop programs with this merging feature, other than `briss`.

3.2 Box Adjustment

I think there are two complications in MuPDF-PyMuPDF main APIs for box processing, and while I tried to mend them a bit, I'm not sure it adds much value, compared to extra trouble.

- MuPDF-PyMuPDF main APIs use float numbers, away from PDF real number strings.
- MuPDF-PyMuPDF reverses y-axis direction, making the bottom of MediaBox, to top.

For example, take the first page of `PDFUA-Ref-2-05_BookChapter-german.pdf` (in PDF/UA Reference Suite 1.1).

It only defines `MediaBox`:

```
[0 0 595.276 841.89]
```

It becomes in MuPDF-PyMuPDF:

```
Rect(0.0, 0.0, 595.2760009765625, 841.8900146484375)
```

When setting cropbox `(10, 10, 585, 831)` by `fitz.Page.set_cropbox`:

```
Rect(10.0, 10.0, 585.0, 831.0)
```

When save to file (note second number, `10.89`):

```
[10 10.89 585 831.89]
```

`pdfslash` adjusts this `0.89`, so running the crop command `'crop 1 10,10,585,831'`:

```
(10, 10.89, 585, 831.89)
[10 10 585 831]
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

C

P