pdfslash

Open Close

Mar 16, 2023

CONTENTS:

Crop PDF margins from interactive cui.

CHAPTER

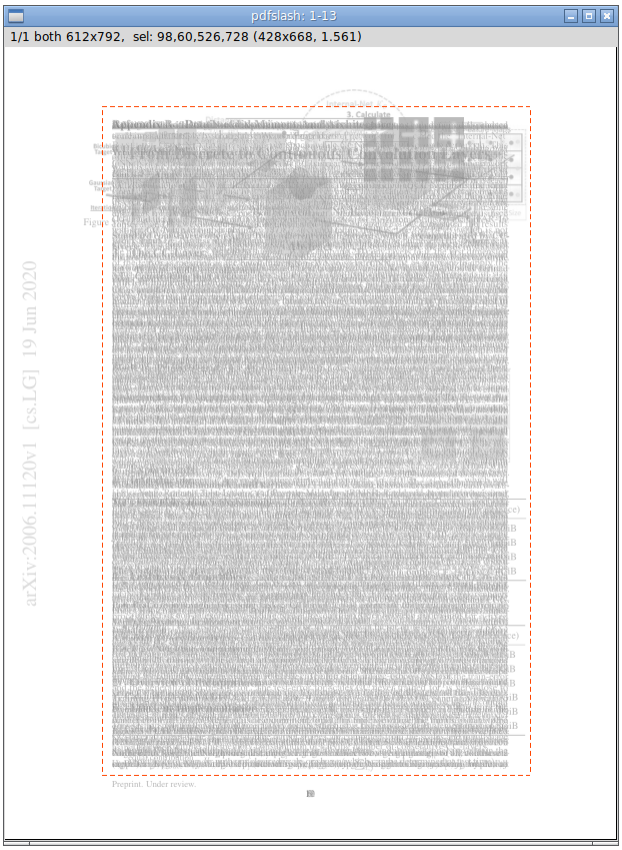
ONE

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).

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

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:

{{ _fromsource_conf }}

2.3 Commandline

{{ _fromsource_commandline }}

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.

{{ _fromsource_nstr }}

- Box syntax is as follows.

{{ _fromsource_box }}

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.

{{ _fromsource_cmds }}

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

{{ _fromsource_gui }}

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.

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