the codex

Life with Linux — A Zine

Typeset in LaTeX

Issue #002

Kenneth John Odle

2021.10.01

Impressum

FYI, this is made in LaTeX using the report document class. It then gets exported to a letterhalf (5.5 in x 8.5 in) pdf, which then gets made into a booklet using PDF Booklet (`https://pdfbooklet.sourceforge.io/wordpress/`).

I'm pushing this to my own git server as I write this. You can find it here: `https://git.kjodle.net/kjodle/the-codex`. New issues will be pushed after they are complete.

You can just skip over all the diversions in here if you want. It's just how my mind works. (And yes, there will be politics in this. *You have been warned.*) Also, I use a lot of em-dashes, parentheses, and footnotes because that is also how my mind works. It's just one big long stream of consciousness up in here most days.

Contents

Chapter 1

The Later Salad Days

Boring, early life stuff when my world smelled like sweat and disinfectant and room temperature bologna. Feel free to skip this. I wish I could.

1.1 The Joy of Commodore 64

1.2 High School Computer Class

Chapter 2

A Scanner Clearly, or More Thoughts on Being an Archivist

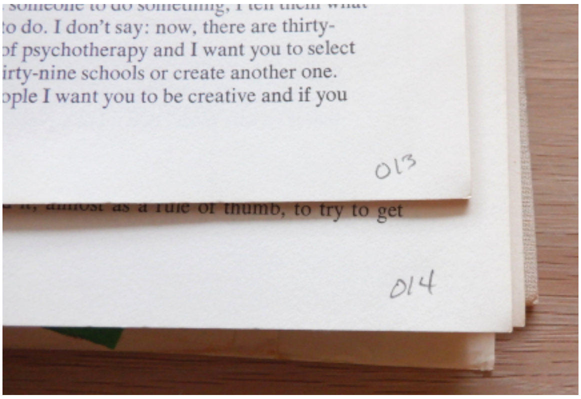
In the first issue of this zine, I wrote about a basic workflow for archiving books through scanning them into pdf files. While I covered about everything I wanted to cover on the computer end of things, I barely talked at all about the physical labor that goes into scanning a book. For those who are interested, here's what happens behind the scenes before you even get to the computer.

2.1 The Non-Computer Stuff

First, you have to cut the book apart, and then separate the pages from the bindings. Older books are generally signature bound, and so it's simply a matter of cutting the backing off the signatures, cutting any strings holding them together, and then separating the signatures. This sounds easy, but it's a lot of work. If the book is perfect bound (i.e., individual pages are glued together, rather than signatures), it's just a matter of separating the pages very carefully a few pages at a time.

Second, you then have to trim the bound edges, so that the pages are separate. Perfectly bound books tend to have glue creeping up between each page, whereas signature bound books tend to have glue only creeping up between the signatures. I have a paper trimmer that allows me to clamp the pages down so that they don't move as I cut them, and I highly recommend something similar. It also has a measured grid to the left, so that I can ensure I'm cutting all the pages to the same width. (**Protip:** Put a piece of painter's tape on the grid to make this even easier.)

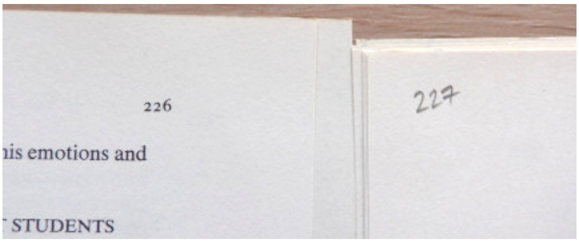
After that, separate your pages into groups of equal numbers of pages that you will scan. This should be however many sheets your scanner can handle easily at one time, and will depend largely on the kind of paper the book was printed on. I generally find ten sheets (i.e., 20 pages) work well, and make it easier for me to count. Smaller groups means more work up front, but it also means that it is easier to fix things when (not *if*) something goes wrong.


to do. I don't say: now, there are thirty-
of psychotherapy and I want you to select
irty-nine schools or create another one.
ople I want you to be creative and if you
013
almost as a rule of thumb, to try to get
014

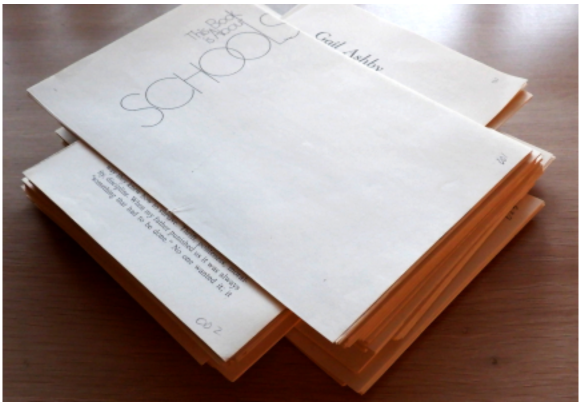
Number all of your groups with the filename they will eventually have. I use a pencil and mark this lightly (or not so lightly, depending on the day) in the lower right corner of the first page:

This is all about workflow for me. Since my scanner (a Brother MFC-J8050DW) scans whatever is facing *down* in the document feeder, after I scan the first (i.e., odd-numbered) side, I should see odd numbers facing up in the ADF. I then know that I need to scan the side that is now facing down, which means that I don't turn them over, I just rotate them 180° in the *xy*-plane.

Most books have unnumbered pages. This should go without saying, but it's one of those things that you don't think about until after it becomes an issue: *number all the blank pages*. Again, I just use a pencil:



Once you do all of this, you're ready to scan. You should have a pile of stuff that looks something like this:

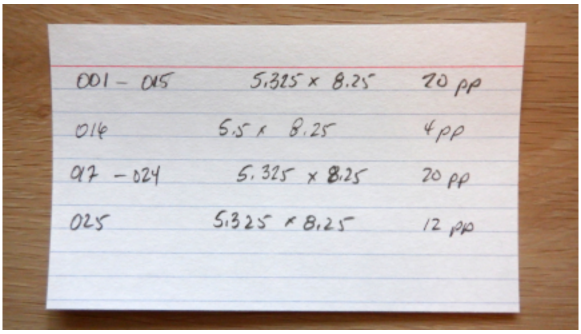


As it turns out, if you have a couple of pages that bleed to the middle[1], you can pretend that they don't and simply trim those pages the same size as all your other pages. Or, if you want to preserve that bleed, you'll have to remove those sheets *before* you trim the edges, and separate them

[1] Generally because of a photograph or illustration that continues across both the left and right pages.

very carefully down the middle. If you are lucky, they are in the middle of a signature, and you can separate them with a sharp knife. If you are not lucky, they will be somewhere else, and may have glue holding them together, meaning you have to very carefully prise them apart. No matter how carefully you do this, you will inevitably lose some data.

You will end up with two sheets (i.e., four pages) that are a different size, and which will need to be scanned separately. In which case, it's good to use a cheat sheet to keep track of which groups are which sizes and how many pages are contained in each. I like to just jot this down on an index card, but if the book you are scanning is complex, you'll need a bigger sheet of paper.



001 – 015	5.325 x 8.25	20 pp
016	5.5 x 8.25	4 pp
017 – 024	5.325 x 8.25	20 pp
025	5.325 x 8.25	12 pp

Now we are *finally* ready to start scanning.

2.2 What Does This Have to do With Linux?

You may be wondering why I am spending so much time talking about using scissors and pencils and rulers when this is a zine about Linux. Sure, this is what I need to do to get ready to scan and put all those scans together using the command line application `pdftk`, but what is the point here?

If you recall back in the first issue, I said that doing things on the command line makes you think about *outcomes*. Thinking about outcomes doesn't matter just on the computer. Like I said earlier, there is no "undo" button in real life.[2] Once you've cut something apart, there's no putting it back together. You *have* to think about what you want the next step of the process to be so that you don't do something in this step that makes the

[2] Although I sometimes think that lawyers are just rich people's undo buttons.

next step afterward difficult or even impossible. You have to think ahead about what you want to end up with. You have to know what you want.[3]

I sometimes think that a GUI is like the menu at the McDonald's drive through.[4] If you are lucky, you are behind the person that knows what they want. They planned ahead. They thought about the outcome they wanted (full stomach, happy taste buds) and chose something ahead of time that would get them to that outcome. But a lot of people (too many people, in my opinion[5] get up to that order screen and *that's* when they decide to start thinking about outcomes. They are so used to seeing a menu in front of them that they can't even begin making a decision without seeing it.

And let's face it: the menu at McDonald's has not really changed in years. Yes, they have new things, but they also advertise the hell out of them when they do. How can you *not* know about their new menu item if you watch more than 30 minutes of television a day? But again, a GUI does not encourage you to think. The command line does. And again: most people just don't like to think.[6] They like the *illusion* of choice, and that is what substitutes for thinking most of the time. "What are you getting at McDonald's?" is too often followed by "I don't know; I'll think about it when we get there."

Of course, if you are thinking about outcomes, chances are you don't eat fast food very often anyway, because the long term outcomes are obesity, heart disease, and hypertension. But damn, those fries are good!

2.3 A GUI Solution

For what it's worth, there is a GUI for `pdftk`. It's called PDF Chain and you can find it at `https://pdfchain.sourceforge.io/`.

Despite all my prattling on about the many advantages the command line has for your brain, I'm not opposed to using a GUI, actually. (I mean, I have Ubuntu installed on two machines and Kubuntu on a third.) A GUI does make life easier in many ways, and what I like about one in a case

[3]And here I readily admit that you are probably going to do this a few times before you figure out a process for getting to that point. The workflow I laid out in section 2.1 took a few books to develop.

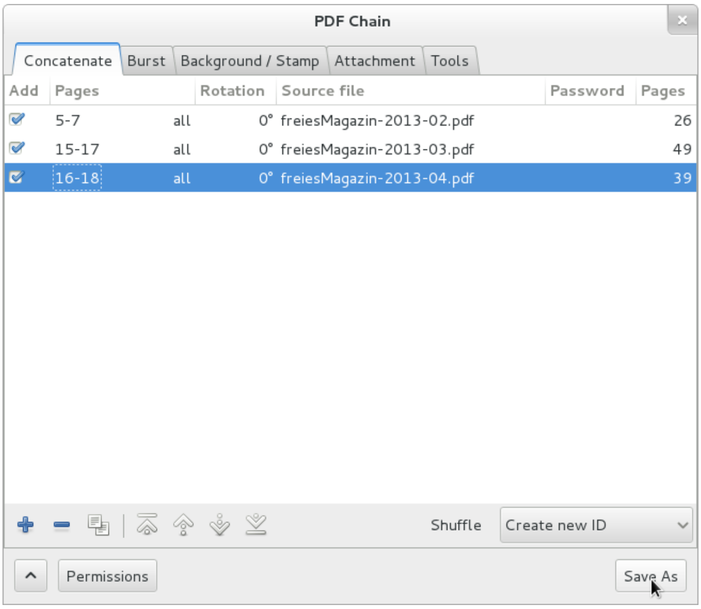
[4]It's no wonder that the menu in a GUI is called a *menu*, when you think about it.

[5]Just one of many reasons I don't eat fast food any more.

[6]I admit, I like to think, but I don't like to think *all the time*. Sometimes my brain needs a rest, so I make some popcorn and pull up something corny to watch on television for a while. But then my brain starts wandering, and I know it's time to get back to work.

like this is that if you're someone who has to manipulate pdf files rarely or only once, it's probably easier to just use a GUI than it is to learn the command line. Efficiency plays a role here, as well. If I'm going to use this all the time, it's definitely more efficient for me to learn the command line approach. But once or twice a year? Or only once ever? A GUI is much more efficient.

tl;dr: If you're only going to use a tool once, there's no issue with using the simplest tool required to get the job done.



Add	Pages		Rotation	Source file	Password	Pages
✓	5-7	all	0°	freiesMagazin-2013-02.pdf		26
✓	15-17	all	0°	freiesMagazin-2013-03.pdf		49
✓	16-18	all	0°	freiesMagazin-2013-04.pdf		39

Create new ID

Chapter 3

Make Life Easier with bash Aliases

I'm not going to get into the difference between the command line, the terminal, and bash (the Bourne Again Shell, if you are interested[7]) For now, let's just assume that you know about `ctrl alt t` opening a terminal window to get you access to the command line.

If you are used to using the command line, chances are that you have a certain set of commands that you use a lot. For example, if you're pushing to your Github repos all the time, you probably are typing `git push origin main` quite often. You can create a bash alias that makes your life a lot easier.

Chances are, you will open the terminal in the root of your `Home` directory. You can tell by typing

```
$ pwd
```

`pwd` stands for "`print working directory`" and shows you where you are. If you're in the home directory, you'll get something that looks like this:

```
/home/username
```

where "`username`" is your login name. If you're *not* in your home directory, you can get there with this command:

[7] You may not be, but after all, you are reading *this* so you very well may be. I'll talk about that in a future issue.

```
$ cd ~
```

The ~ is shorthand for your home directory. (If you are logged in to the terminal as your username, it takes to `/home/username`. If you are logged in to the terminal as `sudo`, it takes you to the `root` directory in the top level directory—where all those Linux directories are that we talked about in the last issue.) We are looking for an invisible file, so execute this command:

```
$ ls -a
```

The `ls` command will list visible files, the `-a` option shows all files, including the invisible ones.

We are looking for a file called `.bashrc`. If you have a lot of stuff installed, you may have to scroll around to find it. Once you find it, open it in a text editor. This is in your home directory, so you can use whatever text editor you want and no `sudo` privileges should be required.

If you want, scroll through it. There is some interesting stuff there. But we want to create aliases, so go to the end, and type

```
# my aliases
```

The # means that this line is a comment. We'll add our aliases beneath this line.[8]

Let's use our Github example. Add the following line:

```
alias gpush="git push origin main"
```

Save the file, and then go to one of your git repos, make some changes, and commit them. Then, when you want to push the changes to the remote repository, instead of typing `git push origin main` just type `gpush`. Your terminal will do the rest for you.

You can also execute bash scripts as well. In addition to a local backup on an external drive, I also backup the directories in my home drive to a remote storage location. To make life easy, I created a script (called, naturally, `backup.sh`) in each of those directories to back them up. To execute those backup scripts, I just need to go to that directory, open the directory in a terminal and type `./backup.sh`.

[8]Let's stop and reflect for a moment on how important it is to add comments to whatever we work on. When you come back to this file in six months or two years and ask yourself "Why is this here? Who wrote this?" your comment will tell you. I often find it handy to include the date as well, and any url where I found something useful. Check the source code of the preamble to this document for examples.

The problem here is that a lot of times, I'm not even in those directories when I'm saving files to them. I'm somewhere else. And to open the directory in my GUI and then open it in a terminal, or to open a terminal and then navigate to that directory, is a little *too* much when I want to run that backup script. Remember, you want to back up soon, and you want to back up often. Backing up on that basis is a good habit to have. So let's remove as many obstacles to that habit as possible.[9] In this case, we'll add an alias to run those backup scripts.

This is what I have in my `.bashrc` aliases:

```
alias kdoc="bash $HOME/Documents/backup.sh"
alias kdow="bash $HOME/Downloads/backup.sh"
alias kpic="bash $HOME/Pictures/backup.sh"
alias krec="bash $HOME/Recordings/backup.sh"
alias ktem="bash $HOME/Templates/backup.sh"
alias kvid="bash $HOME/Videos/backup.sh"
```

Let's look at the first one. `kdoc` is the name of the alias. Since my first name is Ken, I prefix these with the letter `k` so I don't get them mixed up with something else. `bash` means to run this as a bash script. `$HOME/Documents/backup.sh` means "go to the home directory, then go to the `Documents` directory, and run this script called `backup.sh`".

We can do other things as well. Log into your webhost via ssh a lot? Try this one:

```
alias kssh="ssh username@webhost.com"
```

Replace "`username`" with your actual username and "`webhost.com`" with the actual host that you log into.

The next time you want to log into your host, just type `kssh` (or whatever you choose to call your command; the choice is yours as long as it doesn't conflict with a built-in bash command), and it will automatically ask for your password, as it has already sent your username to your host. You've now typed four characters instead of 24. Nifty, huh?

[9] I am often amazed by how often people (myself included) want to form a new good habit (eating more fruit, getting more exercise, etc.) and then put as many things as possible in the way of that habit. Again, it's because we're so used to the old, bad habit that we don't think. Sometimes, we just need to get out of our own way.

This is probably my favorite, though:

```
alias kls="ls -Ahl"
```

This gives us a directory listing, but with these options:

- `A` lists all files and directories, including invisible ones (but excluding the . and .. directories[10]).
- `h` gives us file sizes in human readable sizes (i.e., "4.0K") instead of bytes.
- `l` gives us the listing as a list, because I find that's more readable, especially with a directory that contains a lot of stuff.

Again, I'm typing three keystrokes instead of seven. When you spend eight or more hours a day on the computers, whatever keystrokes you can save really start to add up.

And that's it. Just about anything you type often on the command line can be turned into a bash alias to save you time. Go for it.

[10] If you've ever wondered about what these are, here's a simple explanation. The . (dot) represents the current working directory, i.e., the one that you are in. The .. (dot dot) represents the parent directory, i.e., the directory that contains the directory you are currently in. Whenever you create a directory in a Unix-based system, it is added as a new entry to its parent directory, and these two entries (hard links) are created in the new directory.

`ls` and `ls .` are the same command: they give you the contents of the directory you are in. `ls ..` gives you the contents of the parent directory to the one you are in. It's the same as going up into your parent directory, getting a content listing, and then moving back into the child directory you were just in. And for what it's worth, you can do `ls ../..` to get the content listing of the grandparent directory. Nifty? Depends on how lost you are.

`cd ..` will move you up into your parent directory, whereas `cd .` moves you nowhere, because you are literally telling the terminal to change the directory to the current directory. It's a bit like changing into the underpants you are currently wearing.

Anyway, enough of the stupid pet tricks.

Chapter 4

What Have I Installed?

Chapter 5

What’s to Like About Linux?

I have an app on my phone called “The Stoic” that shows quotations from various Stoic philosophers. As I was working on this issue, this popped up:

> Because a thing seems difficult for you, do not think it impossible for anyone to accomplish.
>
> —**Marcus Aurelius**