

# the codex

Life with Linux — A Zine

Typeset in L<sup>A</sup>T<sub>E</sub>X


Issue #002

Kenneth John Odle

June 28, 2022

## Impressum

All contents ©2022 Kenneth John Odle

Although this is now in your hands, and it's also on the web, so if you really wanted to steal this, I've made it pretty darn easy. I can't imagine why anyone would want to, though. However, you don't need to, because this is licenced under a CC BY-NA-SA 4.0 Creative Commons license. More information is at <https://creativecommons.org/licenses/by-nc-sa/4.0/>. 

FYI, this is made in L<sup>A</sup>T<sub>E</sub>X 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.

The buttons are from the Button Optimizer website, which is here: <https://buttonoptimizer.com/>. I'm not sure if I like this concept or not. We'll have to see. (Note from the future: turns out I'm not keen.)

**Errata:** To err is human, to document those errors is divine. A list of errata can be found at <https://git.kjodle.net/kjodle/the-codex/wiki/Errata>.

**Credit where credit is due:** A lot of people have come forth (mostly from Reddit) to help me out in various ways. See the preamble to this document in the source code to see them. One aspect of our society is that nobody *has* to help you. It is wonderful when it happens, and I am grateful for their help.

The pictures of a Commodore 64 is courtesy of Bill Bertram. It was published at <https://commons.wikimedia.org/wiki/File:Commodore64.png> where you can also find the Creative Commons 2.5 license it was licensed under. I did slightly crop the top and bottom to make it fit better. LaTeX didn't care. The picture of a Commodore 128 is courtesy of Evan-Amos, and was published at <https://commons.wikimedia.org/wiki/File:Commodore-128.jpg>, where you can find the Creative Commons 3.0 license it was licensed under.

# Contents

<b>1</b>	<b>The Later Salad Days</b>	<b>4</b>
1.1	The Joy of Commodore 64 . . . . .	4
1.2	High School Computer Class . . . . .	8
<b>2</b>	<b>A Scanner Clearly, or More Thoughts on Being an Archivist</b>	<b>13</b>
2.1	The Non-Computer Stuff . . . . .	13
2.2	What Does This Have to do With Linux? . . . . .	16
2.3	A GUI Solution . . . . .	18
<b>3</b>	<b>Make Life Easier with bash Aliases</b>	<b>19</b>
<b>4</b>	<b>What Have I Installed?</b>	<b>23</b>
4.1	apt . . . . .	24
4.2	dpkg . . . . .	25
4.3	snap . . . . .	26
4.4	flatpak . . . . .	26
4.5	The Real Issue . . . . .	26
<b>5</b>	<b>What's to Like About Linux?</b>	<b>28</b>
<b>6</b>	<b>Ubuntu 22.04</b>	<b>31</b>
<b>7</b>	<b>Coda</b>	<b>33</b>
7.1	What I Learned About L <sup>A</sup> T <sub>E</sub> X While Creating This Issue . . .	33
7.1.1	L <sup>A</sup> T <sub>E</sub> X Without a GUI . . . . .	34
7.1.2	Custom Page Sizes . . . . .	36
7.2	Why I Love L <sup>A</sup> T <sub>E</sub> X . . . . .	38

# 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

The first computer I ever owned (and thus could use whenever I wanted to, provided it did not annoy the adults in the house) was a Commodore-64. (And yes, my use of this machine seemed to bug adults no end, and I have no idea why. I guess the same adults who thought it was a waste of time playing video games simply thought that a computer is another, more expensive type of video game. The lesson I learned here is to try to get some knowledge before you jump to criticism.)

The “64” stood for 64 kilobytes, which was the amount of memory it had. If you’ve never heard of a kilobyte before, and are wondering how many gigabytes that is, it’s time for some math, and also introductory computer science.

Hold on. You’ve never heard of a *kilobyte*? Wow, either we’ve really moved along, or I’m old, or both. Probably both. My knees hurt in the morning. Yeah, probably both.

Computers are just a series of switches. Each switch is a *bit*. Eight bits make up a *byte*, which is enough memory to remember a single character.<sup>1</sup> Since each bit is just a one or a zero, there are 256 possible different characters you can record in a byte. (Mathematically, it works out to  $2^n$

---

<sup>1</sup>See <https://web.stanford.edu/class/cs101/bits-bytes.html> for more information.

possible combinations. Since in this case  $n = 8$ , then we have  $2^8 = 256$  different combinations of ones and zeroes.<sup>2)</sup>

Time for some math, which looks like this:

$$\begin{array}{rcccccc}
 & & & & 1 \text{ byte} & = & 8 \text{ bits} \\
 & & & & 1 \text{ kb} & = & 1,024 \text{ bytes} & = & 8,192 \text{ bits} \\
 & 1 \text{ MB} & = & 1,024 \text{ kb} & = & 1,048,576 \text{ bytes} & = & 8,388,608 \text{ bits} \\
 1 \text{ GB} = & 1,024 \text{ MB} & = & 1,048,576 \text{ kb} & = & 1,073,741,824 \text{ bytes} & = & 8,589,934,592 \text{ bits}
 \end{array}$$

If you don't trust my math, check out <https://www.matisse.net/bitcalc/>, which is where I did this math. It's actually kind of fun, and they get bonus points for including the source code of the perl script behind this sorcery.

Anyway, I'm driving a laptop now that has 8 GB of memory in it. (And I've seriously considered upgrading it to 16 GB). A gigabyte is equal to 1,048,576 kilobytes. The math (and heck, let's use dimensional analysis because it's fun) looks like this:

$$8 \text{ GB} \times \frac{1,048,576 \text{ kb}}{1 \text{ GB}} \times \frac{1 \text{ Commodore 64}}{64 \text{ kb}} = 16,384 \text{ Commodore 64's}$$

So the computer I'm on now has as much memory as 16,384 of the computer I had when I was 13 years old. If that doesn't seem like a lot to you, I paid \$175 for my current computer (used) in 2016, and paid \$200 (new) for a Commodore 64 in 1981. That's \$3,276,800 in 1981 dollars, which is the equivalent of \$8,651,869.50 in 2016 dollars, respectively. I didn't have three million dollars when I was thirteen, and I certainly don't have over eight million dollars now. Sadly.<sup>3)</sup>

---

### Oh look, it's a diversion

If you've studied the metric system, you know that *kilo-* is a prefix that means a "a thousand" and *mega-* is a prefix that means "a million."

Hold on. (Again.)

1,024 is *not* a thousand, and 1,048,576 is *not* a million. For my en-

tire life, we've just walked right past this and pretended that we didn't notice. Doing science stuff? *kilo* is a 1,000 and *mega* is a million. Doing computer stuff? Then *kilo* means 1,000*ish* and *mega* means a million*ish*. Move along now, noth-

---

<sup>2</sup>To see the actual combinations, visit <https://user.eng.umd.edu/~nsw/chbe250/number.htm>, and to see which characters those numbers translate to, see <https://www.rapidtables.com/code/text/ascii-table.html>.

<sup>3</sup>Check out <https://www.in2013dollars.com/us/inflation/1981> for the actual numbers.

ing to see here.

That discrepancy should bug you. It bugs me, but I also have bills to pay, so when someone asks “*how much* does it bug you?” my honest answer is that I’ve worked *very* hard to not let it bug me all that much.

But still.

Fortunately, it also bugged the International Electrotechnical Commission (an international standards commission whose job it is to standardize things) enough so that they introduced a few new terms in 1998.<sup>4</sup> **Kibibyte** means *exactly* 1,024 bytes, and not one byte more, not one byte less. **Mebibyte** is exactly

1,048,576 bytes. And so on with gibibytes, tebibytes, and pebibytes. They are all some form of  $2^n$ , which means they *accurately* describe just how many bytes we’re talking about here.

The following table is filled with much beauty:

1 kibibyte (ki) = 1,024 bytes

1 mebibyte (Mi) = 1,024<sup>2</sup> bytes

1 gibibyte (Gi) = 1,024<sup>3</sup> bytes

1 tebibyte (Ti) = 1,024<sup>4</sup> bytes

1 pebibyte (Pi) = 1,024<sup>5</sup> bytes

I can at last sleep soundly.

Let’s get back to our story.

I purchased this computer from the back of a K-Mart, in much the same way the men in the small town I grew up in went to the back of the video store to rent porn. I guess it’s fair to say that I lusted after this computer (although a much different form of lust) so the comparison is apt.

Unfortunately, when you bought a computer in those days you got exactly that in the box: a computer. There was no monitor, there was no disk drive, there was no printer. You just got a computer in a box with a power supply. I had scrimped and saved forever to buy



this, and had fortunately also managed to save enough for a monitor, which in those days was a big, heavy cathode-ray tube device (CRT, for short). One of the selling points of the C-64 was that it was portable. You could just pick it up and take it with you. (It seems like all computers in the movies back in the day either took up entire rooms or buildings—think

<sup>4</sup>The standard is ISO/IEC 80000, section 13. This standard is all about the International System of Quantities, and if you’re into that sort of thing, it is utterly *fascinating* reading. I’m not being facetious here—humans measure *everything*, and this document describes how we do it.

Hal-9000 in *2001: A Space Odyssey*—or were something you could carry in your hand—think the tricorders in *Star Trek*. We’ve never managed a happy medium.) The monitor, however, was anything *but* portable. It was heavy, it was bulky, and it was fragile. Slam a lid closed on a modern laptop and everything will probably be fine. Knock a CRT off the table and it’s toast. If it lands on your foot, you’ll probably end up with a broken foot.

The second computer I ever owned was a Commodore 128. Whereas its predecessor only had 64 kb of memory, the C-128, as we called it, had a whopping 128 kb of memory—twice the memory for nearly the same price. While the C-64 could only run in the 1 Mhz<sup>5</sup> mode, the C-128 could run in 1 Mhz mode or 2 Mhz mode, but such were the demands on its resources that running in 2 Mhz mode meant the screen would go blank—it simply didn’t have the power to process that fast and display that processing on the screen at the same time. For comparison, I just pulled up a cheap Lenovo laptop on Best Buy’s website that has an Intel i3 processor with 8 GB of memory and a 256 GB solid state drive and is on sale for \$389. It has a clock speed of three *gigahertz*.



Unlike the C-64, which was round and bulky, the C-128 was slim and sleek. In fact, one of its selling points was its portability, and I recall seeing a brochure where a college student is walking around campus with one under his

arm. What they left out of that illustration, of course, is his friend dragging a little red wagon with a very large and very heavy CRT monitor in it, along with an incredibly heavy power source. They were portable in theory, but not in any practical manner. But compared to a mainframe that took up an entire room or building, it was light years ahead of its time. And I *felt* like that, too, like this computer was going to take me places where I would be light years ahead of where I was then.<sup>6</sup>

Like I said, the laptop I am using now has 8 GB of memory, which is a lot more than 128 kb of memory. But how much more? It’s difficult for most people to visualize numbers, especially when they are orders of magnitude apart, and looking at raw numbers doesn’t give our brains much to latch onto. We need to *visualize* these numbers, which is why I used

---

<sup>5</sup>megahertz—a measure of a computer’s computing speed

<sup>6</sup>It didn’t, because you need so much more than just a computer to get ahead. You need resources, you need people who believe in what you’re doing and support you, and you need people who can point you in the direction of the next step. If you don’t have those things, you’re not a visionary with a bright future ahead of him, you’re just a nerd with a computer.

money earlier. But because I like math and most people like food, let's visualize these numbers a different way: through food.<sup>7</sup>

Let's assume that those 128 kb of memory are equivalent to one happy childhood meal. Maybe it's your dad cooking out on the weekend, or your Nonna making homemade meatballs, or maybe it's not a childhood meal; it's just you and your significant other sitting on the futon enjoying spaghetti Lady and the Tramp style. Pick whatever meal you love, that you think you could eat every day. That meal is the equivalent of those 128 kb of memory.

We've already established that there are 1,048,576 kilobytes in a single gigabyte. Let's do some more math.

$$\frac{1,048,576 \text{ kb/Gb}}{128 \text{ kb/meal}} = 8,192 \text{ meals}$$

That means that for every single gigabyte of my modern laptop's memory, you could eat that meal once a day for 8,192 days. Since there are 365 days in a year, we'll do the math again:

$$\frac{8,192 \text{ meals}}{1 \text{ Gb}} \times \frac{1 \text{ day}}{1 \text{ meal}} \times \frac{1 \text{ year}}{365 \text{ days}} = 22.44 \text{ years/Gb}$$

I hope you really like that meal, because you're going to be eating it every day for the next 22.44 years. Oh, but wait, that's for *one* gigabyte. My laptop has eight. A bit more math tells me that you'll be eating this meal for  $22.44 \times 8 = 179.6$  years. Oof, not only is it tasty, but it's also life-extending.

## 1.2 High School Computer Class

I went to high school in a very small,<sup>8</sup> very rural, very midwestern small town. My graduating class was 55 people, 51 of which actually managed to show up to graduation. (I showed up to graduation with a black eye, but that's another story for another zine. Also, our class president got on a plane the day after graduation, flew to California, and moved to a pot farm. Progress. That's the story, anyway.) We only had two high school science

<sup>7</sup>A&W once tried to launch a 1/3 pound burger to compete with McDonald's quarter-pounder, the idea being you get more meat for the same price. But it didn't sell because people thought a third of a pound was less than a fourth of a pound. (Again, the metric system for the fucking win.) They could get  $4 > 3$ , but they couldn't understand  $1/4 < 1/3$ . See <https://awrestaurants.com/blog/aw-third-pound-burger-fractions> for the full story.

<sup>8</sup>We had one stoplight when I left there in 1990. I visited a few years ago, and they are now up to three stoplights. Progress.



teachers: Mr. Fusko, who taught all the biology classes and Mr. Dick, who taught chemistry, physics, and advanced mathematics.<sup>9</sup>

Mr. Dick also taught computers.

This was the mid 1980s, and my very small, very rural, very redneck-laden high school had a computer lab. In reality, it wasn't an *actual* computer lab. It was just a room where they took all the student desks out, replaced them with chairs and tables, and plopped computers on them. Our high school had been built in the 1960s, when computers took up an entire wing of a building. Nobody thought to design a classroom with more than four outlets in those days, so there were extension cords everywhere.

Still, it had computers in it, and that's what counts.

I don't know if I actually was able to take a computer class with Mr. Dick or not. I remember spending a lot of time in that room, but as a nerdy kid with a crap family and not much of a social life, it's only natural that I would have migrated here during my free time before school, after school, during lunch hour, or whenever I decided to skip class (which was often) or whenever I wasn't leading a meeting of the Millard Fillmore Fan Club.<sup>10</sup> I remember that this room had a *lot* of Trash-80s in it, with their dark silver metallic cases and black highlights, but it also had five or six CP/M machines in it, with their off-white cases and their neon green displays. (Does anybody remember CP/M? Does anybody remember laughter?<sup>11</sup>)

Anyway, if I did take a computer class with Mr. Dick, it was a very pale experience compared to my seventh grade experience,<sup>12</sup> because I don't remember it at all. No doubt there was a curriculum<sup>13</sup> for the class, complete with quizzes, assignments, tests, and long lists of vocabulary to memorize. It would have been completely dreadful and I would have hated it, and would have naturally forced all remnants of it from my memory, so that I would have room for lots of terrible memories from lots of terrible jobs.

Here's the flip side though. Let's assume I *didn't* have a computer class

---

<sup>9</sup>His real name; I'm not kidding. His first name was Joe, which means that I learned a lot of chemistry, physics, and advanced mathematics from Joe Dick. He was the nicest guy in the world, though. I can't remember the name of the math class I had with him my senior year. It was probably Finite Math, but who knows? I don't remember anything about matrices, but it damn sure wasn't an accounting class.

<sup>10</sup>See my other zine, *just13* issue #3 for more details

<sup>11</sup>You may remember this from a Led Zeppelin song, but I always remember the version by *Dread* Zeppelin. It's one of the few cases where I feel the cover is better than the original.

<sup>12</sup>See the first issue of this zine.

<sup>13</sup>Not a bad thing when well done, but years of teaching experience have proven to me that most curricula are largely designed to take any and all fun out of learning, to say nothing of teaching.

with Mr. Dick. In that case, what I have are very fond memories of a room that I should not have had access to but was welcome to. For a very narrow slice of the space-time continuum, there a place where I was welcomed and people were okay with me just wandering around and pushing buttons.<sup>14</sup>

Considering the huge problem that the modern tech industry faces with inclusivity and diversity, I feel pretty lucky to have had that opportunity. We need to create more of these places. Maker spaces are becoming a thing, and I hope that they are as warm and inviting as I remember that high school computer lab to be. Tech should open a lot of doors for a lot of people who normally find themselves locked out of opportunities, but tech is *expensive* and these costs are gatekeepers in too many instances.<sup>15</sup>

If Oprah were really interested in changing the world, instead of handing out cars to a few hundred members of her studio audience,<sup>16</sup> she should be handing out Raspberry Pis and Arduinos to hundreds of thousands of poor kids. But her viewers understand cars, not Raspberry Pis and Arduinos, and Oprah does not do what is good for society, but what is good for her bottom line. (The real privilege in getting a free car from Oprah is that you are able to actually take time off from your job and your life and go to Chicago to attend a taping of her show. But nobody really thinks about *that*.)<sup>17</sup>

It saddens me that we are now at a price point where technology should be able to transform the lives of millions of people, and free them from the situation they are in now. I know a lot of people say that kids are so much more comfortable with technology now, but this really isn't a great thing. When I was a kid, you used technology to change your life. There wasn't a lot of technology, so it basically boiled down to learning how to use an extremely slow computer to do tasks you'd rather not do, using a VCR to record shows so that you could watch them at a later time, and duplicating cassettes and creating mix-tapes. As I grew into young adulthood and started teaching, I saw kids doing exactly that.<sup>18</sup> Mix-tapes were replaced by mix-CDs. You could manipulate technology to improve your life, and

---

<sup>14</sup>I have for a very long time wanted to write a science fiction story about an alien race who loves pushing buttons so much that they build devices that are essentially just buttons to push that do absolutely nothing. Imagine a PlayStation controller without the actual PlayStation.

<sup>15</sup>Which is one reason that I am interested in getting a Raspberry Pi and playing around with it, but in this case, I lack time, not money.

<sup>16</sup>Watch it while it lasts: <https://www.youtube.com/watch?v=pviYWzu0dzk>

<sup>17</sup>You want to know who does a lot of good for people, but doesn't make it all about herself? Dolly Parton, that's who. Dolly Parton is a saint. We truly don't deserve her. Forget Zod. Kneel before Dolly.

<sup>18</sup>Of course, these were the kids whose parents had the money to buy a computer and pay for access to the internet, neither of which were cheap in the early 90s.

when you were done, you shut it off and called it a day.

Now it's the other way around: technology manipulates you and you **can't** shut it off. Websites (I'm thinking of Amazon here, as they are the best at it, but plenty of other websites do this as well) now *tell* you what you want to buy. You can buy things on subscription so that you don't have to think any more.<sup>19</sup> Streaming services control what you watch or listen to. No longer can you just walk into a record store or a video store and get what you actually want. No longer will people experience the serendipity of walking into a store and finding a movie or album that becomes a huge part of their life. (I've discovered some of my favorite books that way. I truly believe certain objects just call to you.<sup>20</sup>)

People have become addicted to their phones in a way I'd never imagined possible. (If you are ahead of me at a red light, and the light turns green, but you don't go because you don't notice that the light is now green because you're looking at your phone—I *will* let you know that the light is now green and you can proceed through the intersection. Believe me, you *will* know.)

Advertisements are everywhere, on every app, on every streaming service.<sup>21</sup> They are constantly telling you that you need this product or this service, and it has become very difficult to screen that out, so much so that even drunk purchasing is now a substantial part of the economy.<sup>22</sup> Before, you had to leave your house to get manipulated into buying something, now you don't even have to leave the house. You can literally shop yourself out of house and home without ever leaving your house or your home.

What was promulgated as a potential servant, ever willing and able to come to our assistance, has now become our master. For more about this, I've created a YouTube playlist that you can watch here:

<https://kjodle.info/techincharge>

What strikes me most about some of the videos in the the playlist I've linked above is that the emphasis on using technology is always that it

---

<sup>19</sup>It can be devilishly tricky to actually unsubscribe from some of these things, to the point where it's easier just to absorb the expense (we can always get the kids vaccinated *next* year) and just decide that this is how we live now.

<sup>20</sup>I know that's kind of woo-woo, and I am not into a lot of woo-woo stuff. Maybe it's more just a matter of getting out of the house and walking around with our eyes open.

<sup>21</sup>Including mindfulness meditation apps—see the back cover.

<sup>22</sup><https://www.finder.com/drunken-shopping>, <https://www.marketwatch.com/story/amazon-is-prime-territory-for-drunken-shoppers-2019-03-25>, <https://www.techtimes.com/articles/240241/20190326/drunken-us-adults-spend-48-billion-shopping-online-and-amazon-is-so-happy-about-it.htm>.

will free up our time to spend time improving ourselves and relaxing. It's amazing how in many ways, we've far surpassed the technological capacities we imagined earlier, but rather than freeing up our time to improve ourselves (in true *Star Trek* style), we are merely slaves to our own creations. I shudder to think how "Hotel California" has become a daily reality for so many of us. But do any of us really have more spare time as a result of technology today? (I would like to think that as a species we are intelligent enough to do that, but the fact there there is a patch of plastic the size of Texas floating around in the middle of the Pacific Ocean tells me otherwise. "Out of sight, out of mind" is meant to be a warning, not permission to shift blame.)

It needn't be that way, however. If more of us had been able to experience what I experienced in that seventh grade computer class, perhaps our relationship to technology today would be different. Perhaps we would realize that, like the wheel or fire, technology is a tool to help us improve our lives, not to be led around by it. Of course, look at what we've done with both wheels and fire—we are destroying each other and ourselves as well with them. Perhaps we never should have climbed down out of the trees onto the grassy savanna.

I don't know. It seems that our philosophy, or at the very least, our ethics, surrounding technology and what we do with it always lags far behind the cutting edge of that technology. If there are other species in the universe that have managed to travel faster than the speed of light, certainly they *must* have come to a philosophical decision about the role of technology in their daily lives.<sup>23</sup>

---

<sup>23</sup>Which explains why we only get visited by the butt probe aliens now. All the other aliens in the universe have given up on us.

## 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 just 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.

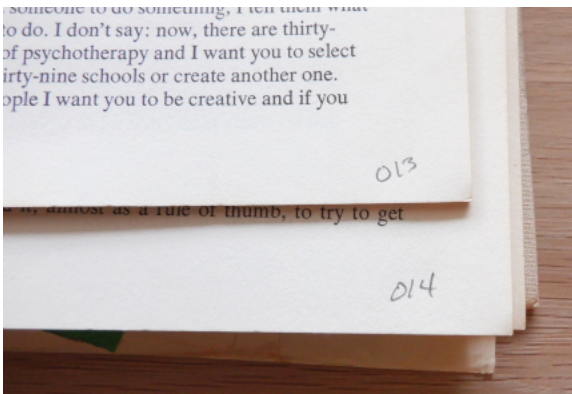
Second, you then have to trim the bound edges, so that the pages are separate. Perfectly bound books tend to have glue creeping up between each page, whereas signature bound books tend to have glue only creeping up between the signatures. I have a paper trimmer that allows me to

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.

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:

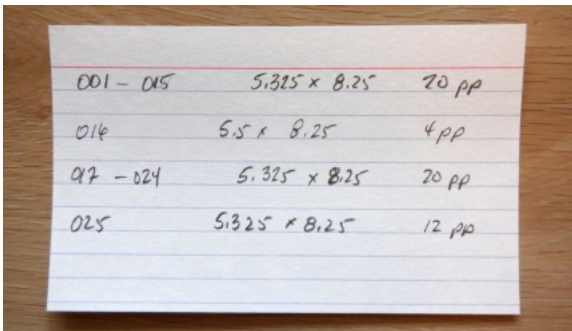


As it turns out, if you have a couple of pages that bleed to the middle<sup>24</sup>, you can pretend that they don't and simply trim those pages the same size as all your other pages. Or, if you want to preserve that bleed, you'll have to remove those sheets *before* you trim the edges, and separate them

<sup>24</sup>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.



You can do this math as a check on the final project.

001 – 015 = 15 × 20	300 pages
016 = 1 × 4	4 pages
016 – 024 = 8 × 20	160 pages
025 = 1 × 12	12 pages
Total	476 pages

After you are done scanning and combining files, you can open that final file.pdf and you should be on page 1 of 476 pages.

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 pdf tk, 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.<sup>25</sup> 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 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.<sup>26</sup>

I sometimes think that a GUI is like the menu at the McDonald's drive through.<sup>27</sup> 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<sup>28</sup> 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.<sup>29</sup> 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!

---

<sup>25</sup>Although I sometimes think that lawyers are just rich people's Undo buttons.

<sup>26</sup>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.

<sup>27</sup>It's no wonder that the menu in a GUI is called a *menu*, when you think about it.

<sup>28</sup>Just one of many reasons I don't eat fast food any more.

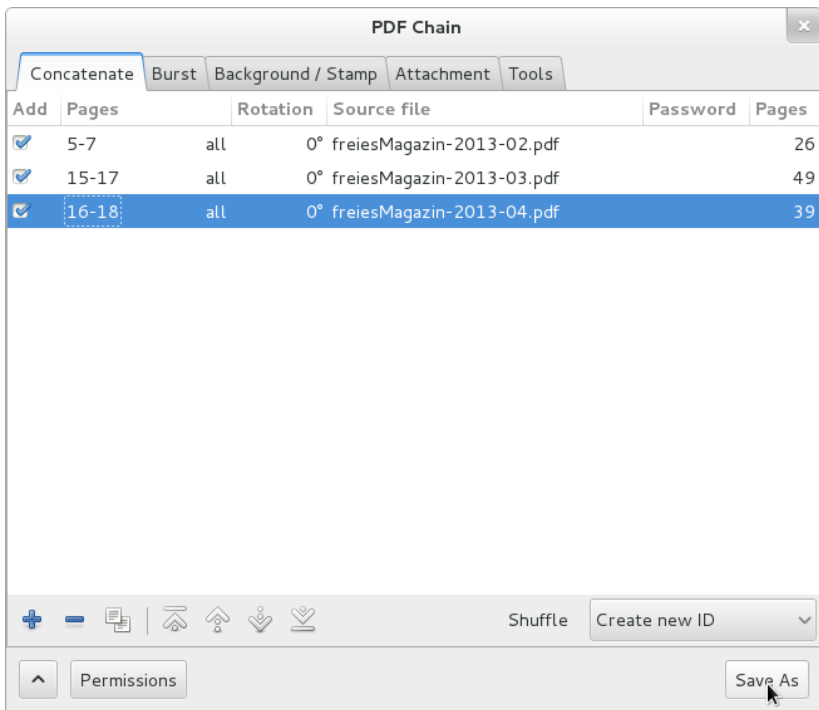
<sup>29</sup>I admit, I like to think, but I don't like to think *all the time*. Sometimes my brain needs a rest, so I make some popcorn and pull up something corny to watch on television for a while. But then my brain starts wandering, and I know it's time to get back to work.

## 2.3 A GUI Solution

For what it's worth, there is a GUI for pdf tk. 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—all GUIs that make Linux easier to use.) A GUI does make life easier in many ways, and what I like about one in a case 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. There's no point in being a command-line ascetic.



## Chapter 3

# Make Life Easier with bash Aliases

Level: Intermediate <sup>30</sup>

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.<sup>31</sup>) 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
```

---

<sup>30</sup>Why is this here? See the chapter “What's to Like About Linux?” later on in this issue.

<sup>31</sup>You may not be, but after all, you are reading *this* so you very well may be. I may talk about that in a future issue. (Also, it's called that whether or not you are interested in that. I know of very few things that have a conditional name.)

where “username” is your login name. If you’re *not* in your home directory, you can get there with this command:

```
$ 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.<sup>32</sup>

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

---

<sup>32</sup>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.

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`.

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.<sup>33</sup> 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 kdown="bash $HOME/Downloads/backup.sh"
alias kpics="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?

---

<sup>33</sup>I am often amazed by how often people (myself included) want to form a new good habit (eat more fruit, get 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 -Ah1"
```

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

- A lists all files and directories, including invisible ones (but excluding the . and .. directories<sup>34</sup>).
- h gives us file sizes in human readable sizes, i.e., “4.0K” instead of “4096 bytes”.
- l gives us the listing as a list, because I find that to be more readable, especially with a directory that contains a lot of stuff. I’d rather just scroll up and down than scroll up and down *and* scroll right and left.

Again, I’m typing three keystrokes instead of seven. When you spend eight or more hours a day on the computer, 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. It’s a great way to manage your mischief.

---

<sup>34</sup>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 human tricks.

# Chapter 4

## What Have I Installed?

I've mentioned the Unix Principle before: each application should do one thing and do it really well. The advantage to this principle for end users is that you don't need to worry about installing (or even paying for) a huge app that does a million things when you only need it for doing one. The advantage for developers is that you don't *need* to develop an app that does everything, so you can keep a laser-like focus on making an app that works really well.<sup>35</sup>

The downside is that you are likely to install a lot of apps as a result. This is ordinarily not a problem, unless you are going to migrate to a new machine or are swapping out an HDD for an SSD. This is where I found myself in 2020 when Ubuntu 20.04 came out, and I decided to install a new SSD in my 2013 Asus laptop, replacing the 300 GB HDD it came with. Of course, I could simply clone the old HDD onto my new SSD and then swap them out and upgrade, but for reasons that are long, complicated, and very likely boring, I decided this would be as good a time as any to just start with a clean slate.

Again, normally this is not an issue. But I have a Behringer C-1U microphone that I use for podcasting, and it only works on Ubuntu if I have a particular piece of software installed. For the life of me I could not remember what it was, but it was installed on this laptop, so it ought to have been easy to find.

For what it's worth, any software package you install that has a graphical user interface will appear in your applications menu. (On Ubuntu, it's the

---

<sup>35</sup>Although there will always be those people who get your app for free and then complain that it *doesn't* do everything. Hey bub, Swiss army knives are two aisles over and come with a price tag.

“Show Applications” button in the lower left corner of your main screen.) But a lot of apps don’t have a GUI—they’re meant to be used from the command line. There isn’t a graphical way to see a list of those. So we’re off to the command line.<sup>36</sup>

## 4.1 apt

Here’s the command:

```
$ apt list --installed
```

When I ran this on my newly upgraded version of Ubuntu 22.04 (Jammy Jellyfish) its output ran to 2,811 lines, starting with

```
accountsservice/jammy,now 22.07.5-2ubuntu1 amd64 [installed,automatic]
ac1/jammy,now 2.3.1-1 amd64 [installed,automatic]
acpi-support/jammy,now 0.144 amd64 [installed,automatic]
acpid/jammy,now 1:2.0.33-1ubuntu1 amd64 [installed,automatic]
adduser/jammy,now 3.118ubuntu5 all [installed,automatic]
```

and ending with

```
zim/jammy,jammy,now 0.74.3-1 all [installed]
zip/jammy,now 3.0-12build2 amd64 [installed,automatic]
zlib1g-dev/jammy,now 1:1.2.11.dfsg-2ubuntu9 amd64 [installed,automatic]
zlib1g/jammy,now 1:1.2.11.dfsg-2ubuntu9 amd64 [installed,automatic]
zstd/jammy,now 1.4.8+dfsg-3build1 amd64 [installed,automatic]
```

So what’s happening here? Let’s take a look at the relevant part of the `man apt` file:

`list` is somewhat similar to `dpkg-query -list` in that it can display a list of packages satisfying certain criteria. It supports `glob(7)` patterns for matching package names as well as options to list installed (`-installed`), upgradeable (`-upgradeable`) or all available (`-all-versions`) versions.

This command is giving us a lot of output because it’s showing *everything* that was installed using the `apt` command. You might think that you haven’t really installed that much stuff, but `sudo apt install` will sometimes install additional software needed to make the software you are actually interested in run. If you ever install something using `apt` and see a message like “The following additional packages will be installed:” those additional packages will appear when you use this command.

---

<sup>36</sup>**Note:** What I’m describing in this section applies to Ubuntu and its derivatives, since that’s what I use and most familiar with. Other flavors of Linux will probably have similar ways of doing what I’m going to talk about here.



Remember those glob patterns? If you want to check to see whether a particular app is installed, you can pipe the output to `grep` and let it do the work. For example, let's see if I have `ghostscript` installed:

```
$ apt list --installed | grep ghostscript
```

and because I do, I get this output:

```
ghostscript-x/jammy,now 9.55.0~dfsg1-0ubuntu5 amd64 [installed,automatic]
ghostscript/jammy,now 9.55.0~dfsg1-0ubuntu5 amd64 [installed,automatic]
```

`apt` also has a related command, called `apt-mark`:

```
$ apt-mark showmanual
```

But more about that in a bit.

## 4.2 dpkg

If you want something a bit more tabular, you can always run

```
$ dpkg-query -l | less
```

You will get a nice, very *wide* table that shows you a list of all installed packages, their version, their architecture (`amd64` and `whatnot`), and a short description. Unfortunately, the output is again huge. Outputting this to a text file produced 2,971 lines.

That short description is pretty handy however when you're rooting around in the `bin` folder and wondering what things do, actually. If you want to find out what a particular package does, you can again pipe its name to `grep`:

```
$ dpkg-query -l | grep ghostscript
```

which gives us this:

```
ii ghostscript 9.55.0~dfsg1-0ubuntu5 amd64 interpreter for the
PostScript language and for PDF
ii ghostscript-x 9.55.0~dfsg1-0ubuntu5 amd64 interpreter for the
PostScript language and for PDF - X11 support
```

I don't find this very useful however, because again, that is formatted in a far wider table than most terminal windows (I took out a lot of extra space, and still had to wrap things onto a second line), and honestly, if you want to know what an app does, it's probably easier to just look it up on the web. (Still, it's rather useful to know that one of these packages is for X11 support.)

### 4.3 snap

If you use snap packages (which are controversial in some quarters), you can always see what you've installed by using this command:

```
$ snap list
```

This is again displayed in a pretty wide table, but it also gives you the Version, the Revision, the Tracking status (“latest/stable” in most cases), the Publisher (Canonical, Mozilla, etc), and Notes.

### 4.4 flatpak

If you use flatpak to install software, the command is similar to snap:

```
$flatpak list
```

That's all I have. Sorry, but I have just haven't gotten around to using flatpaks. It's possible I never will.<sup>37</sup>

### 4.5 The Real Issue

The real problem with all of these approaches is that none of them do a very good job of telling what you actually want to know: which software packages did I *deliberately* install? And that's because computers aren't smart. (They also aren't stupid. They just *are*.) They only know what you tell them. They have no idea what you *mean*.

In fact, I found a page on StackExchange<sup>38</sup> that was first asked in December 2010 and was last modified in March 2022, has 24 answers, and has been viewed 4.6 million times. There simply isn't a way (that I could find, at least) to figure this out easily. Even something as straightforward as `$apt-mark showmanual` still shows a lot of programs that I *technically* installed myself, simply because I updated from 20.04 to 22.04 via the command line.

---

<sup>37</sup>The Candian comedy troupe The Kids in the Hall have a song called “The Daves I know”. Look it up on YouTube, and when it gets to the part about Dave Capisano, the mechanic (“I hardly know him”) that's where I am with flatpak.

<sup>38</sup><https://askubuntu.com/questions/17823/how-to-list-all-installed-packages>

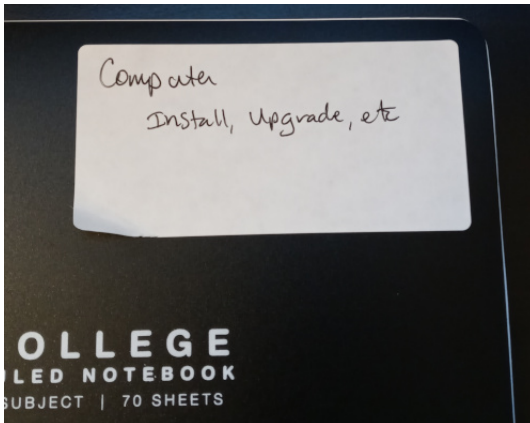
*Meet my friend, Occam.*

The irony is that the Unix Principle came out in the 1970s, but in the early 1960s the US Navy came out with a different, yet similar, principle called KISS: keep it simple, stupid.

This means that instead of looking backward (i.e., “What did I install a year ago when I was several beers deep?”) we should be looking forward (i.e., “What am I about to install, and then possibly remove, while I am completely sober and trying to solve a problem that is currently driving me up the wall and so once I’ve solved it, I will be so happy to have solved it that I will immediately forget what I did to solve it?”).

The solution is simple: *write it down.*

When I upgraded this computer, I also bought a new desk for it, and I threw a notebook in the drawer. Whenever I install something, I make a note of it in the notebook. If I delete it a few days or months later, I add a note of when I deleted it and why. (That “why” is important. I may be able to remember the problem that particular piece of software solved, and may not remember the rat’s nest of problems it went on to create for me.)



*Of course I would select a black notebook and a black desk.*

*Let's use **all** the toner.*

Sure, you could just keep a list on your computer (text files if you want to keep it simple, or you could just <shudder>use a spreadsheet as a database</shudder>. But if your computer fails, then your list fails. Sometimes paper really is the best way to go.

Of course, I’ve also just purchased a RocketNotebook and have been experimenting with that, and maybe storing my scribbles in the cloud is the way to go. We shall see.

# 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

When I first thought about doing the second issue of this zine, I thought it would be really cool if I included some sort of key to any tutorial that I include, that tells you if this is a beginner level tutorial, an intermediate level tutorial, or an advanced level tutorial. If you peek at the beginning of chapter three, you'll see a button that looks like this:

Level: Intermediate

Like it? I have a couple more:

Level: Beginner

Level: Advanced

But the more I thought about it, the more I began to think that this is **not** a good idea. There are a couple of reasons for this.

First, who am *I* to decide if something is beginner, intermediate, or advanced level, especially when it comes to computers? I'm not an expert in all things computers, much less all things Linux. The most I can say is that “Oh, yeah, this is one of the first things I learned how to do” which *implies* that it's a beginner level. But it might not be. It's just where *I* started. Somebody else might start somewhere else, in a place that I didn't bother

to get around to for years, and which I tend to think of as intermediate or advanced level. But to them, it's where they started, and so they'll also think of it as beginner level. LaTeX has been around for a long time, but it always seemed very advanced to me. As it turns out, it's not difficult to learn. (Although like most things, learning a thing and mastering a thing are very different activities.) Things can be difficult if you don't have the proper tools for them, but does that mean they're advanced?

Second, these words are highly contextual. What does "beginner" even mean? Someone who's just sat down to a computer for the first time in their life? Or someone who's been working on computers since they were ten years old, but are just now beginning to learn LaTeX? These words are easier to define in other areas, such as sports or music, where you have to learn to skate before you can play hockey, and you have to learn scales and chords before you can play music.<sup>39</sup> Computers are the great levelers of walls and fences, provided, of course, that you have access to one.

Third, people may self-select out of things. Many years ago, I taught computer classes at my local community education center.<sup>40</sup> It was a tremendous amount of fun, both for me and my students, who came from all walks of life.<sup>41</sup> They were *occasionally* intimidated by certain things, but for the most part, they were eager to learn as much as they could.

The class was extremely inexpensive, but it was rarely ever full, which made me wonder why more people didn't take this class. There are two answers here. First, this was the early 90s and a lot of people didn't even have computers at home, so what would be the point for them? (You don't need a car if you have no place to go.) Second, a lot of people were terribly intimidated by computers, because they thought you had to be really smart to use them. So they self-selected out of something they were perfectly capable of learning because they thought it was too advanced for them.<sup>42</sup> Let's encourage people to self-select *in*, rather than out, because once they

---

<sup>39</sup>Although I am not an expert in either of these things, and maybe it is possible to learn hockey and music without learning how to skate or learning scales and chords first. To each their own.

<sup>40</sup>One of the things I learned is that is a right way and a wrong way to teach people how to use computers. I should write about that sometime.

<sup>41</sup>Or as many walks of life as one could find in this one-stoplight town.

<sup>42</sup>People are really afraid of making mistakes and looking stupid, but that's a by-product of our public education industrial complex. You can't learn *anything* without making mistakes. In fact, people who are truly experts about things often point out that they learn as much or more from their mistakes than they do from their successes. But we have turned making mistakes into something to be ashamed of, rather than something which will help us learn and understand things. Fuck that attitude. Go forth and make mistakes. (Just not in the voting booth.)

get here they'll have a lot of fun. Let's not be gatekeepers.

Fourth, it implies that there is a hierarchy, which I hate. There are many ways in. Because math is so strongly allied to computer science, we tend to view learning about computers as hierarchical as well.<sup>43</sup> But that just isn't the case. I started learning how to write BASIC when I was in sixth grade because that's all that was available to us. But basic BASIC is no different than basic Fortran, or basic Cobol, really. The only difference is that BASIC is fairly limited in scope, which was, sadly, all that was deemed appropriate for kids. It's like giving kids little fake plastic tools when they really want to build something. Put the kids in a sandbox with real tools and turn them loose: let them experiment with them and figure out how they work. But we never do that.<sup>44</sup>

I don't want to be a gate-keeper. I want to be a gate-opener. In fact, I want to be a crusher of gates. There is no single path. You have to find your own path. Find your mentors, and carve your own trail. This isn't about ego. Knowledge is not pie. There's plenty for everybody.

---

<sup>43</sup>I think this is one of the reasons that most people aren't good at math and don't like math. Sure, you need to know *some* algebra to do geometry, but you don't need to be an algebra expert. If you really like geometry, and are encouraged to apply yourself to it, you'll eventually learn all the algebra you need. The same is true of trigonometry and even calculus. You don't need to become an expert in those earlier forms of math; you only have to become good enough at them to move on to the next step. People who write math curricula should take note, but they won't.

<sup>44</sup>This is, coincidentally, the same sort of thinking that doesn't like sex education, despite the many studies that have shown, conclusively and repeatedly, that when kids have access to high-quality, non-biased sex education, the rates of teen pregnancy and teen STDs decrease, often dramatically. (It's silly. A little learning can help you know what to do, but it can also help you know what *not* to do. Sure, some people need to burn their hand to learn that the stove is hot, but the vast majority of us can learn just by watching others howl in pain from a burned hand. Pictures are worth a thousand words. Moving pictures are worth a million words.)

# Chapter 6

## Ubuntu 22.04

I've been with Ubuntu for a while now, and generally get pretty excited about new releases. Ubuntu 22.04 was no different. Until I installed it, that is.

Ubuntu 22.04 is without a doubt the most bug-filled LTS<sup>45</sup> version of Ubuntu I have ever used.

First, they have removed Python2. If you have any applications which depend on Python2, such as PDF Booklet,<sup>46</sup> those tools are now unusable. And despite numerous attempts to reinstall Python2 just to make this one app work, I have thus far been unsuccessful.

Second, there is this weird problem with saving files from *any* browser. You can pick a folder to download to, and from that point on in your session, this is the only folder you can download items to. Sure, you can migrate to a different folder and try to download there, and it looks like you're actually saving your file there, but nothing gets downloaded. You can look at the folder and at your download history and quite plainly see that there is nothing there.<sup>47</sup> As it turns out, if you click on the *filename* in the save dialogue box (and therefore give it focus), then you can change where you download the file to.

This is an odd bug to allow to get through, as for the longest time the question "What does everyone use Internet Explorer for?" was answered by

---

<sup>45</sup>Long Term Support

<sup>46</sup>Which is what I use to make the physical form of this zine. See <https://pdfbooklet.sourceforge.io/wordpress/> for more information.

<sup>47</sup>See <https://askubuntu.com/questions/1406265/ubuntu-22-04-firefox-does-not-download-file-to-desktop> and <https://askubuntu.com/questions/1406674/cant-download-files-after-upgrade-to-22-04>

“To download an actual web browser.” I guess you can always use Ubuntu to download a version of Linux which is not plagued by such obvious mistakes.

Third, the default video player (Totem) cannot play .mp4 files.<sup>48</sup> Yes, I know that .mp4 is not strictly open source (as some of the codecs are patented) and I also know that you can install these codecs with `$ sudo apt install ubuntu-restricted-extras`, but if you have a phone, or a video camera, they probably record video as .mp4 files, and it would be nice if you could see view them on your computer.

Fourth, the new screen capture tool in the Gnome desktop is an absolute disaster. It requires you to go through a “screenshot portal” each and every time you create a screenshot, *regardless* of which screenshot tool you use. It’s ridiculous, as I (and many other users) already had a workflow for screenshots, and this is not acceptable in an open-source operating system (which actually makes me wonder how “open-source” open-source software actually is). Of course, people are upset, because this is a solution in search of a problem to solve. I have no idea why *anybody* thought this was a good feature to include. And of course, the developers have closed off any discussion of this topic.<sup>49</sup> Listen, if a lot of people complain, you should listen to them. If a lot of people complain, and complain vehemently, then it’s probably because you fucked up really badly.

I’ve lived and worked with Ubuntu for several years now. And until 22.04, I’ve always been extremely happy with each new release and upgraded as soon as it was available. And I’ve always upgraded without regret. I guess that is now a thing of the past.

Look, I get that you want to keep to a release schedule. But we had a global health crisis in the middle of that. I would have been okay if you had said “Yeah, mate, we get you’re more concerned about your family’s well-being. We are too. So this release is going to be six months/a year/two years/what-the-fuck-ever behind.” I would have been *so* okay with that.

But I’m not okay with Ubuntu 22.04. It is an utter embarrassment. How it was ever released without consideration of these issues is beyond my comprehension. I used to recommend Ubuntu to all my friends, and now I don’t. I just can’t. I myself am embarrassed about these issues. Do yourself a favor and avoid Ubuntu 22.04 until these issues (and no doubt other issues I’m not aware of) are addressed and fixed.

---

<sup>48</sup>See <https://askubuntu.com/questions/1406254/after-installing-ubuntu-22-04-the-default-video-player-is-unable-to-play-any-vi>.

<sup>49</sup>See [https://gitlab.gnome.org/GNOME/gnome-shell/-/merge\\_requests/1970](https://gitlab.gnome.org/GNOME/gnome-shell/-/merge_requests/1970), <https://gitlab.gnome.org/GNOME/gnome-shell/-/issues/4895>, and <https://github.com/fl1atpak/xdg-desktop-portal/issues/649>.



# Chapter 7

## Coda

### 7.1 What I Learned About L<sup>A</sup>T<sub>E</sub>X While Creating This Issue

As a big part of the reason I created this zine was to learn more about LaTeX, I'm keeping up with this running list.

1. Need a little horizontal space? Use `\hphantom{<stuff>}` where `<stuff>` is any standard unit. (I use this down below to separate the two images with borders when they are on the same line.)<sup>50</sup>
2. Need a box around an `\includegraphics[scale=•]{•}` item? Just wrap it in `\frame{}`. (Ditto.)
3. Want a blockquote? Use the `quote` environment. (I wrapped mine in a `small` environment to help set it off, as most blockquotes or indented quotations use a slightly smaller font in traditional printed material.<sup>51</sup>)
4. You can draw with the `tikz` package. You can also draw chemical structures with the `chemdraw` package. I have no idea how to write about those things on paper in an interesting way, so it may be some

---

<sup>50</sup>There is more information on spacing at <https://latexref.xyz/Spaces.html> and also at <https://tex.stackexchange.com/questions/74353/what-commands-are-there-for-horizontal-spacing/74354>.

<sup>51</sup>Yes, it bugs me when people use the word *quote* as a noun, but the usage is here to stay, so I shall learn to live (somewhat begrudgingly) with it.

time (or never—never is always an option) before I get around to that. But there’s an example at the end.

5. You can also draw just using the `picture` environment.<sup>52</sup>
6. As with most things that \*nix-based, there is usually more than one way to get to where you are going. Often, there are many ways, and they lead you down paths you hadn’t even imagined. A little research goes a long way. (See the next two sections as examples of this. I had not even thought about this before I sat down to write this.)
7. There are ten characters that have special meaning in  $\text{\LaTeX}$ .

To typeset:

$$\& \% \$ \# \_ \{ \}$$

you have to prepend them with a backslash (`\`).

To typeset:

$$\sim \wedge \backslash$$

you have to use the macros `\textasciitilde`, `\textasciicircum`, and `\textbackslash`.

### 7.1.1 $\text{\LaTeX}$ Without a GUI

Despite my blathering on about the benefits of the command line, I’m actually using a GUI editor called Texmaker (which you can find at <https://www.xmlmath.net/texmaker/>). This seemed the easiest way to learn  $\text{\LaTeX}$  at the time, because all you have to do to get a readable pdf is to press F1.

But you don’t need to go that route. You can do this entirely from the command line. Simply create a  $\text{\LaTeX}$  document in any text editor, and save it with a `.tex` extension. From that point, simply run the following command in a terminal:

```
$ latex file.tex
```

This command should generate the following files:

---

<sup>52</sup>There is a good tutorial at [https://www.overleaf.com/learn/latex/Picture\\_environment](https://www.overleaf.com/learn/latex/Picture_environment).

```
file.dvi
file.aux
file.log
```

`file.aux` contains information your document needs to manage any cross-references in your document. `file.log` contains information about how your file was processed; if you run into errors, this is a good place to find a solution, or at least to find what to search the internet for. But it's the `file.dvi` file that we're interested in.

`.dvi` files are device independent files. They're a lot like PostScript or PDF, but *without* font embedding. To convert this to a pdf file, run the following command:

```
$ dvi2pdf file.dvi
```

This should generate `file.pdf` which you can read in any document viewer. You may need to install `dvi2pdf`—on my system (Ubuntu 20.04 at the time) it was not installed.

You can also just run `pdflatex` (which again, you may have to install), which skips over making a `.dvi` file:

```
$ pdflatex file.tex
```

This should generate the following files:

```
file.aux
file.log
file.pdf
```

I have noticed that when I generate the pdf file using the former method, I get a much smaller file than I do the second time. As an experiment, I ran the `integral.tex` file that I created later in the next section through both of these methods. Running the file through `latex` and then through `dvi2pdf` resulted in a pdf file that was only 7.0 kb in size. But when I ran it solely through `pdflatex`, I ended up with a pdf file that was 30.5 kb big. This is most likely due to a difference in compression methods<sup>53</sup> so this could make a difference for you if you are working with large documents.

Go forth and manage your mischief.

---

<sup>53</sup>See this for more information: <https://tex.stackexchange.com/questions/38145/why-does-pdflatex-produce-bigger-output-files-than-latexdvi2pdfm>

## 7.1.2 Custom Page Sizes

Okay, this is important enough that it deserves its own section.

Part of what makes L<sup>A</sup>T<sub>E</sub>X great is that it's really good at typesetting mathematical formulas, such as

$$x^n + y^n = z^n$$

Here's the thing, though: I didn't create that formula in this document. It's just an image. I created it in a separate LaTeX document, using a custom page size. This can be handy if you want to use it in something that doesn't typeset math formulas, such as a presentation.<sup>54</sup>

The source code looks like this:

```

1      \documentclass{article}
2      \usepackage[
3          left=0.1cm,
4          right=0.1cm,
5          top=0.1cm,
6          bottom=0.1cm]
7      {geometry}
8      \begin{document}
9      \pdfpagewidth=2.3cm \pdfpageheight=0.7cm
10     \noindent $ x^n + y^n = z^n $
11     \end{document}

```

That's it; that's the entire document. Let's take a closer look at what is happening here.

Lines 2-7 use the `geometry` package to give us some pretty tight margins. This is a good thing, as this is going to be clip art. We could set them to zero if we needed to (and which might not be a bad idea, actually).

Line 9 is where the magic happens. It allows us to set the actual page size of this example. And yes, I could have just used the `geometry` package to declare the page size in the preamble. It's what I do with this document. But doing that affects *all* the pages in our document. This lets us handle page size on a page-by-page basis, as we shall see. All we need to do is add a new page and resize everything again. Take a look at this:

```

1      \documentclass{article}
2      \usepackage[

```

---

<sup>54</sup>PowerPoint much? As much as I try to avoid PP, it seems to have gained something akin to Favored Nation Status in the business world. Such is life, alas.

```

3         left=0.1cm,
4         right=0.1cm,
5         top=0.1cm,
6         bottom=0.1cm]
7     {geometry}
8     \begin{document}
9     \pdfpagewidth=2.3cm \pdfpageheight=0.7cm
10    \noindent $ x^n + y^n = z^n $
11    \newpage
12    \pdfpagewidth=4.6cm \pdfpageheight=1.4cm
13    \noindent $ x^n + y^n = z^n $
14    \end{document}

```

I admit, I had to play around with the variables here, which is a bit of a pain. Fortunately, there is a way to automatically fit the page size to the content, provided I only want to create a single page: use the standalone document class. This source code:

```

1     \documentclass{standalone}
2     \begin{document}
3     \noindent $ x^n + y^n = z^n $
4     \end{document}

```

gives us this output

$$x^n + y^n = z^n$$

We've now managed to do with one line of code what previously took us 8 lines of code. I would call that efficient.

You'll also notice that there is no border spacing around the second formula. This is handy in the event that I want to drop this into a word processing document. I'll add a box around these images so you can see the actual size:



Also, for reasons I don't know yet, the typical way of starting and ending a math environment in LaTeX (i.e., `\[...]`) doesn't work in the standalone document class.. Only `$. . .$` and `\begin{math} . . . \end{math}` do.

The standalone class is definitely pretty handy. Now, let's combine this idea with converting .tex documents directly into pdf files without using a GUI. There is a program called `dvipng` which you should be able to install from the command line, which will convert these .dvi files to .png files just by running:

```
$ dvipng file.dvi
```

If you need a .gif file instead,<sup>55</sup> just add the `--gif` flag:

```
$ dvipng file.dvi --gif
```

I'm not going to forget about the first method, though. This could be handy if I wanted to create something (such as a business card) that is a standard size that I want to repeat, or if I want to print on a smaller, non-typical format that LaTeX doesn't have a built-in page size for. I have a few ideas where I might use this; I'll try them out and report back in a later issue.<sup>56</sup>

The obvious advantage here is that it's possible to create a document where every page has a different size. You can use the `\parbox{}` environment to more precisely control where text and images are placed on the page.

Is this useful? As stated, that's a loaded question, because it's missing two parameters: *to whom* and *in which context*. This isn't useful to me at all right now, but I can imagine in the future somebody might find a way to make an interesting zine in this way and I like that idea, because that's where technology (i.e., LaTeX) and art (zines) intersect, and this is the most comfortable part of that particular Venn diagram for me.

For what it's worth, I've added a repo of these experimental files to my gitea instance. You can find it at <https://git.kjodle.net/kjodle/codex-latex-experiments>.

## 7.2 Why I Love L<sup>A</sup>T<sub>E</sub>X

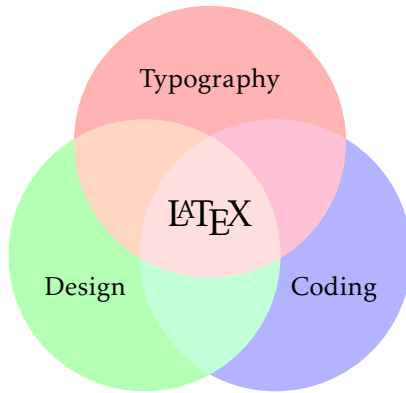
I didn't draw the picture on the next page. It's an example I got from <https://texample.net//tikz/>. I sometimes find it easier to learn a thing by finding examples and playing around with the parameters to see what they do.<sup>57</sup>

---

<sup>55</sup>dude, wtf?

<sup>56</sup>Probably, but no guarantee. Other people have a train of thought; I have a Roomba of thought. I may bump into a chair leg at some point and go off in a completely different direction and forget I ever said this.

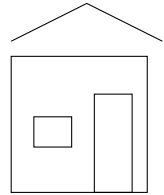
<sup>57</sup>If you are interested in drawing in LaTeX, be sure to check out <https://texample.net/tikz/resources/>



I am very comfortable living in the center of that Venn diagram.  
 (And yes, I played around with that code to make it a bit smaller—as one does.)

Anyway, off to the right is the kind of thing I've figured out how to draw using `tikz`.

Yeah, I've got a ways to go. (If you couldn't tell, it's a house with a floating roof. If you're wondering why the roof is floating, I am too. Let's just assume it's some modern Swedish design.<sup>58</sup>) I literally have a dozen browser tabs open just to draw that little Swedish house, and yes, that is how I tend to learn the best: here's the basic idea, here are a bunch of examples (some of which seem to contradict one another), and here's my sandbox where I play around with it until I get it just the way I like it.



Now that I look at my code, I realize why my roof is floating. Here's my original code:

```

1 \begin{tikzpicture}
2   \draw (0,0) rectangle (1.8cm, 1.8cm);
3   \draw (1.6,0) rectangle (1.1cm, 1.3cm);
4   \draw (0.8,1) rectangle (0.3cm, 0.6cm);
5   \draw (2,2) node{} -- (1,2.5) node{} -- (0,2);
6 \end{tikzpicture}

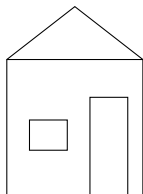
```

Lines 2-3 draw the rectangle, and line 4 draws the triangle. Apparently, I resized the house and forgot to resize the roof. If I change line four to this:

<sup>58</sup>I'm good with this. Let's not make it weird.

```
\draw (1.8,1.8) node{} -- (0.9,2.5) node{} -- (0,1.8);
```

We now have a house with a proper (i.e., non-Swedish modern) roof:



**Word of advice:** only change one variable at a time. Anyway, time to stop playing and get back to work.