the codex

Life with Linux — A Zine

Typeset in LaTeX

Issue #001

Kenneth John Odle

Published on 2021.10.10

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 Boomaga (`https://www.boomaga.org/`).

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.

The image of Linus Torvalds on the front cover is courtesy JericoDelayah from the WikiMedia Commons. The image is over here:

`https://commons.wikimedia.org/wiki/File:4_RETAT_04_Linus_Torvalds.jpg`

You can also find a link to the Creative Commons CC BY-SA 3.0 license there, as well.

The image on the back cover is one that I highly agree with. We built it, it's ours, and we shouldn't be charged for using it. I want my tax dollars to serve my needs. I don't want my tax dollars used to make rich white old men richer.

(If you're reading this zine online in pdf form you won't see this cover, because it's not in this repo. You can check the `.gitignore` file. If you *really* want to see it, you'll need to buy a paper copy, which means we kill a tree and I get validation and a few bucks, and you get a picture of Linus Torvalds that you can see online for free anyway, if you are motivated enough. Weird, huh?)

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 Early 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 Calculators

Before computers were in my life, there were calculators.

These days, every kid has to have an expensive graphing calculator for school starting with middle school math. Specifically, it has to be a Texas Instruments graphing calculator, because the examples in the textbook are all described in terms of a Texas Instruments calculator.

I mean, *sure* you can get your kid that Casio, which has all the same features and all the same buttons and is an order of magnitude cheaper, but you spent all that money on an expensive pre-school, and all that money on expensive tutors. Do you *really* (he asked snottily) want to risk little Jimmy's chances of getting into Harvard because you were temporarily too cheap to buy the right calculator? Just buy the TI already!

Oh my, a diversion already.

A little off track here, but this begs two questions: 1) Why is it always a TI calculator that's required, and 2) Are we teaching kids to learn math or to learn how to operate a calculator? The answer to the first question is that Texas Instruments and the Major Textbook Pub-

lishers™ have colluded to produce expensive books that need to be replaced every two to three years [thereby costing the school district money] and that require expensive calculators[1] [thereby costing you as a parent money]. It's a racket, but that's capitalism for you.

The answer to the second question is that we are teaching kids how to use calculators. Teaching them how to do actual math would require thought on both the parts of the teachers[2] and the parts of the students, not to mention on the parts of parents and especially of administrators, who would also be required to grow a spine—and learn how to use it. Again, education in the United States has become a racket, but that's capitalism for you.

Alas.

(You can probably tell what my thoughts are on the dominant economic system on planet Earth. There *will* be more of that. If you're okay with that, I'm okay with that, too. If you're not okay with it and you want your money back, it's too late—I've already spent it.[3])

I have noticed that even little kids are required to bring little kid calculators to school with them in most of the local school districts. As I write this, the school supply buying season is coming to an end, but for the past six weeks every store was filled with school supply lists and yeah, you have to have a calculator to get into the second grade.

Ironically, the earliest calculators I can remember seeing (not getting my hands on, because they didn't belong to me) were Texas Instruments calculators. I don't remember a lot about them, but an uncle had given a pair to two of my cousins. They took a *ton* of batteries, had red LEDs for outputs (meaning they glowed in the dark—you could use them in the dark if you memorized the keypad), and they were designed for students because they had a go-back-through-all-your-steps-to-see-where-you-done-screwed-up-boy function, which would be a useful feature on modern cal-

[1] A few years ago, I bought a scientific calculator at the **dollar store** and tested it against my very expensive TI-92. It was just as accurate as the more expensive calculator, and cheaper by two orders of magnitude. Did I mention that this is a racket? I really should do a YouTube video or blog post about this.

[2] To be fair, a lot of teachers would like to teach kids how to do actual math. But they also need to eat and when it comes down to the difference between doing what is right and doing what pays the bills, they will do the latter. It's not their fault, really; it's just that the system does not like anybody who sticks out. Keep your head down and the worksheets graded—that's what the system rewards.

[3] But that's capitalism for you! *Caveat emptor!*

culators to learn math, but again, we're not interested in kids actually learning how to think and do something as radical as math.

The other early calculator I remember was a Casio calculator and it was on a watch. A kid I knew for a short time had one, and even let me wear it for a while. (I wish I could remember his name, because this was a tremendous kindness on his part.) I swore that when I grew up, I would own one of these watches.

I grew up and I didn't buy one of them, even though they are still available. I could never justify spending the money on what is—let's face it—just a bit of full-frontal nerdity when there were bills to pay. Nope, just could never bring myself to do it.

It's just me now, and my expenses are numerous but small, and a couple of years ago my local all-in-one-store had all their watches on sale for 40% off, including the name brand watches. I checked—it was in stock. At $25 bucks it was a lot, but on sale it was only $15. I could do this! So I picked it up and looked at it lovingly, thinking about all the good times we would have together as we went forth and explored the world one simple calculation at a time.

But there was a problem. A rather large problem, actually. The print on those buttons is tiny. And my eyes are bad. I couldn't actually read any of the buttons. I use reading glasses when I'm reading or working on the computer, but I don't need them out in the wild. I could wear the watch with me everywhere, but unless I were at my desk, I wouldn't be able to actually use it.

Back on the shelf it went. I'm not going to spend money on something that is not actually useful to me.

At this point, my only hope is that maybe my eyes will get so bad that I'll need bifocals all day, every day. When that happens, will this watch be on sale for so little money ever again? I highly doubt it.

1.2 Speaking of Watches, Timex Used to Make Home Computers

My earliest memory of a computer in somebody's home is of being in an aunt's apartment, where she had a Timex Sinclair hooked up to her television.

I don't remember much about it, actually, other than it was small and sleek and very modern-looking. I do remember that I was not allowed to touch it.[4]

This is where memory gets wonky, because I remember seeing this when I was about ten years old. But according to Wikipedia, the Timex Sinclair[5]was released in 1982, when I would have been 13 or 14 years old. So it's entirely possible that my memory is losing track of *when* things happened, or it's possible that this aunt had some other home computer that for whatever reason my brain thinks is a Timex Sinclair. Who knows? I certainly don't, and I'll probably never find out for sure.

1.3 The Joy of a Trash-80

One thing I'm quite sure about is that in seventh grade a select group of smart kids from my class were allowed to go to the local "skills center"[6] one day a week (Wednesday afternoons, as I recall) to study computers. This was the first time I'd ever laid my fingers on an actual computer keyboard.

Oh look, another diversion.

"TRS" actually stands for "The Radio Shack," as in *The Radio Shack 80*. This program had a room full of TRS-80 Model IIIs, with an integrated keyboard, and, if I recall correctly,[7] two integrated 5.25" floppy disk drives. I loved Radio Shack both because the first computer I was ever allowed to sink my teeth into was a Trash-80 and because for a while there in my youth, it was a tinkerer's paradise.

Those of us with fond memories of Radio Shack, and what it used to be, bristle at the memory of how it was terribly mismanaged at the end

[4]This aunt bought things not because she found them useful, but because other people didn't have them and she wanted to always have a status symbol to point to. I don't remember her actually doing anything *useful* with this computer.

[5]It was called the Timex Sinclair because this was a collaboration between the Timex Corporation and the Sinclair Corporation. I imagine Sinclair handled the R&D and manufacturing and Timex handled the marketing. If so, Timex didn't *technically* make a computer, but they wanted us to think that they did. Good enough for me.

[6]This was a centralized school where eleventh and twelfth graders who definitely weren't going on to college could take classes like agriculture and welding. We used to teach these classes in each school under the guise of "vocational education" but somehow lost our way.

[7]But I probably don't.

of its life. But in some weird post-modern way, Radio Shack does live on, just not as you might expect.

A brief history shall ensue:

Radio Shack was founded in 1921 by Theodore and Milton Deutschmann, two brothers who wanted to cash in on the burgeoning ham radio field. Initially successful, the company was nearly bankrupt in 1962, when it was acquired by the Tandy Corporation. If you've ever been in 4H, that name may ring a bell. Tandy was a leather goods corporation, and had been selling supplies for home leathercrafters since 1919. If you did leathercrafting as a 4H kid, chances are the introductory tool kit and all the materials you needed to make that belt or wallet, came from Tandy Leather Company.

Tandy actually ran Radio Shack fairly well until the mid 1990s, and had many electronic products with a "Tandy" name on them. But in 2000, they decided to drop the Tandy name altogether, and became the RadioShack (one word) Corporation. They sold the leather crafting assets to a company called The Leather Factory in the fall of that same year, and it still operates under the name of Tandy Leather. (If you're into making things from leather, you may want to look them up.)

RadioShack decided to move away from electronic supplies and electronics and focus on selling cell phones, which was a ridiculous move, since everybody else was also selling cell phones. The period from the early 2000s forward was a period of decline, both in sales and morale (not to mention much clenching of teeth by regular shoppers—I went in to buy some screws to repair a laptop and they didn't even carry metric screws at that point), and finally resulted in bankruptcy in 2015.

The name still exists, because other corporate entities bought the rights to it. So there are still RadioShack stores out there, but they're not Radio Shack. The Radio Shack that so many of knew and loved (it was my favorite store to visit at the mall when I was a teenager) is gone and will probably never come back. I mourn the Radio Shack of my youth as I mourn a long lost lover.

That Wednesday afternoon experience was a real game changer for me. It was the first time I had an opportunity to sit down at a computer for an extended period of time and actually accomplish something, rather than just tinker around with the keyboard. We could save our work on cassette tapes until next week, which meant that our projects had some perma-

nence, although I'm certain that all those cassette tapes are either buried in the depths of a storage room somewhere at the skills center, or more likely are buried deep in a landfill somewhere.

This was the golden age for my generation for computers. These days you can buy a computer magazine and it has a CD or DVD with programs for you to try out. (Although I haven't been in a bookstore since the pandemic started, so that may have changed.) In the early 80s, computer magazines had programs *printed* in them, so if you wanted to try out a program, you had to very laboriously type it in, and then spend the rest of the evening debugging it before you actually got to spend the last 15 minutes before bedtime playing around with it.

Some of the TRS-80s did have disk drives. But these were 5.25" floppy drives, not the 3.5" floppies in the hard plastic case. (Heck, we even had a CP/M machine with an 8" floppy drive, if I remember correctly.) You had to load TRS-DOS from a TRS-DOS disk, and then swap it out for a disk that you were going to load a program from or save your work to. I actually remember thinking at one point that if you had a computer with six or eight of these drives stacked up, you would never have to swap out a floppy[8]. You could just start it up and go and never have to worry about it.

Needless to say, when I found out about hard disk drives, my mind was blown.

And yes, in the early days, computers did not have a hard disk drive. I am writing this on a fairly ancient Asus laptop with 8GB of RAM and a 3rd generation Intel i5 chip in it. I bought it used, and the minute I got it, I wiped the drive and installed Ubuntu. That made it speedy, at least a lot faster than it was running Windows. But a while ago I decided to upgrade from Ubuntu 18.04 to Ubuntu 20.04, and I removed the hard drive and replaced it with a two terabyte[9] SSD (solid state drive). This sucker *moves*.

Twelve year old me's head probably would have exploded.

What I loved the most about working on those old TRS-80s was the sense of control that I had, at a level I had never experienced before. When you're a kid, there's a lot that is beyond your control. When you're a poor kid of color in a one-stoplight town, there even more that you can't control. You lack a lot of the agency that better-off, less brown kids have.

But for three hours every week, I could be in control. All of our programs were written in BASIC (TRS-BASIC, if I recall correctly) and if

[8]The name of my next band—Swapping Floppies.

[9]Did I *need* 2 TB? No, I did not. But reader, I got an excellent deal on it.

something didn't work, it was up to me to figure out what was wrong with it. There was nothing wrong with the computer, of course. It only did what I told it to do, and when I told it to do something that made no sense or that it couldn't understand it simply threw up its hands and gave me an error message. [10]

Frustrating? Yes, it *was* frustrating, until I realized that every mistake I made was also an opportunity to get better at writing code. It was an astounding amount of freedom.

That was partly because there was no textbook and no curriculum. The instructor, Fred, was always available to help, to guide, to encourage, and to answer questions, but he mainly left us to our own devices and never told which direction to go in. He left it completely up to us. He was the ultimate travel guide: pick a destination, and he would show you on the map where it was and how much water was in your way. If you needed to know how to use a kayak, he was happy to help, and he didn't care if you got carried away on a side-quest and never got to your original goal. It was a completely open learning environment and it was utterly *amazing*.

Of course, such things can never last long.

I'd hoped to repeat this opportunity in the eighth grade, but it wasn't an option, and I never found out why. I imagine that somebody somewhere decided kids this young didn't need to learn anything about computers, because they were either a fad or the entire thing was just so much "communist computer clap-trap". (This was a very small, very conservative town, after all. More about that next time.)

So eighth grade was back to the grindstone of multiplying binomials, memorizing endless (and pointless) historical dates, and dodging bullies in the hallways. Our PE teacher was an ex-marine who began and ended each class with military drills, so the joy of IF THEN GOSUB was replaced with TEN-HUT! LEFT FACE! RIGHT FACE! MARCH! In one short summer, I had come full circle.

I wouldn't get my hands on an actual computer again until eleventh grade. But that's another story.

[10] This mindset is a good one to have, and has saved me hundreds of hours of troubleshooting things. Rather than assuming the computer is in the wrong, I generally assume that I've told it the wrong thing. What was the last thing I told it? Ah, *there's* the problem. As someone who has helped numerous people with their computer problems, I can assure you that 95% of all computer problems are either PEBKAC (Problem Exists Between Keyboard And Chair) or PICNIC (Problem In Chair, Not In Computer). Of the remaining problems, 4% are 1D-10-T errors, and the last 1% is an actual computer problem, which is usually solved just by rebooting. Such is life.

Chapter 2

What's to Like About Linux

I could go on and on here, but I'll try to keep it short. I can always come back to this. (And I probably will.)

2.1 Control...and an Opportunity

What I like—not love (when it comes to computers, love is about aesthetics for me)—is that I'm in control.

Partly, that's the nature of open-source computing. If you want to know how something works, you can look at the source code. If you don't understand the source code, you can research how the source code works. You can ask questions. (Thank you, StackExchange!) You can do some more research and then learn how to ask *better* questions. There is always something to learn, and once you've learned everything there is to learn about a particular piece of software [11] you can fork it and start contributing to the project yourself.

Wondering how something in Windows works? So is everybody else. There is nothing more frustrating than googling a problem in Windoze, getting hundreds or thousands of results, and every result is just somebody else asking the same question and not getting an answer—just a million other voices crying into the wilderness saying "I have the same problem. Please help."

[11] Which is never really true. What I really mean is that when you've learned everything *you* want to know about it.

And yeah, you can write code and create applications for Windows, and you can solve a lot of problems that way, but you can never make Windows itself better. It is what it is, and if you don't like it, the feature that bugs you might be made better in the next release, or it might be made worse. It's a crap shoot, really.

For what it's worth, Mac OS X, even though it is based on Unix/Linux (I forget which—I dropped out of the Mac world at OS X version 4), is the same way. There *might* be an answer, and there *might* be a solution, but you just *might* be on your own there, buddy.

That's the key when you're working with something that open-source: every problem is an opportunity for you to learn something. You might be able to find a workaround, or a fix, or even realize that you're doing something wrong, and that's why you're having a problem. Who knows? Keep studying and trying things out and you might find an actual bug and be able to contribute a patch that fixes it. That will never happen when you use Windows or Mac. Never.

Linux rewards study in a way that macOS and especially Windows do not, and never will.

2.2 Knowledge is Power

You know what I really, really like about Linux?

The command line.

I've already mentioned this earlier (and I'm sure that I'll probably write about this some more later), but my experience with computers goes back way before Macintosh made the mouse popular (and alas, necessary). You turned on the computer, and there was just this dark screen with a blinking cursor. If you wanted to make it do something, you had to *know* something. With a GUI, you can guess. You can guess a lot, actually, and just poke around all you want because most GUIs come with an undo feature.

There is no "undo" on the command line.

I need to get that on a t-shirt.

Why? Because the command line is like real life. There is no undo button in real life. GUIs have made us lazy—lazy at thinking, lazy at figuring things out. Just do it: if you don't like it, just Ctrl-Z. Just throw that document away and leave it in the recycle bin. If you decide you want/need it later, you can just drag it on out of there.

With a GUI, that "undo" button is always an option.[12] But in real life, you can't *unmake* a mistake. Sure, you can *recover* from a mistake, but you are going to have to do some scrambling, my friend, and if you are at least halfway intelligent, you will definitely think twice about trying that again, or at least trying it *that way* again. You don't want to jump through all those hoops again, so you think about your end goal and try to develop a better workflow for next time.

The command line, in short, makes you think. It makes you plan, it makes you think about the end goal, it makes you remember past failures. The command line makes you think about *outcomes*.

A GUI only makes you think about the next step. Surely all the steps after that will be obvious, *n'est ce pas*? I've seen a lot of people ask questions online where they just want to be told which button to push. They are asking about how to cross the street when what they really want to do is get across town. They are asking for *information* when what they really need is *knowledge*.

Sadly, as individuals and as a society, we are drowning in *information* when what we are starving for is *knowledge*.

2.3 The Unix Philosophy

The Unix Philosophy was originated by Ken Thompson (one of the creators of Unix, upon which Linux is based) and basically says that each program should do one thing and do it well. (There is more to it than this; if you are interested, you can always google it.[13])

This runs counter to physical life, where everything has to be a Swiss army watch. Watch any ad for a new kitchen gadget and this device does *everything* except walk the dog and take out the trash. If it *actually* did all those things and did them well, I would be happy to own one and more than happy to pay a couple of hundred dollars for it.

Unfortunately, it seems that it's impossible to build a device that will

[12] Except for the rare occasion when it isn't. Those times are fun.

[13] Searching for something on the internet is *always* an option these days, and so many people seem to be unable to do just that. Honestly, this is the kind of stuff that gets my underpants in a twist.

Question: "Where can I find *X*?" Answer: The same place I would find it: At the other end of a google search.

Better question: "Which is the **best** source for *X*?" Ah, *now* we have the basis for a discussion. I'll put the kettle on and we can talk about it.

do a large number of things really, really well. I like to cook and so my parents inevitably give me a cooking-related gift every Christmas and every birthday. One year, I received a mandoline-type device—if you've spent any time watching the Home Shopping Network[14] I'm sure you've seen them. It's basically a plastic tray with different cutting inserts, a handle to hang on to the food item, and a box that snaps on to the bottom to hold whatever you are slicing.

I absolutely *love* this thing for slicing potatoes, and since I spend each autumn and winter making scalloped potatoes or au gratin potatoes, it sees a lot of use during those months when the days are short. It does a fantastic job of slicing potatoes into a uniform thickness and it does it far more quickly than I can do it with a knife.

It also includes inserts to make waffle slices (if you rotate the potato 90 degrees on each pass, you're supposed to be able to make waffle fries), inserts for dicing onions, and so forth. But here's the thing: as great as it is at slicing potatoes (and also carrots, which have the same general hardness as potatoes), it does a terrible job at slicing anything else. Basically, the thin and the thick slicing inserts work well for potatoes and carrots, and all the other inserts don't work at all for them, and any other vegetable just doesn't get cut or gets crushed because you have to hold onto it so firmly.

I don't know how much my parents spent on this thing, but if it's anything north of $20, that's a lot of money for something I can already do fairly easily (and actually enjoy doing) with a sharp knife. Don't get me wrong—I love the thing (even though it's a bit of a pain to clean), but if we had spent at least as much time and money engineering the thing as we did marketing it, we might have concluded that it would probably be better to just encourage people to buy decent knives and then teach them how to sharpen them and use them properly.

(Also, I'm not picking on the Home Shopping Network[15] because every company does this. Monty Python, all those years ago, even had a skit about this, which you can find if you google "simpsons individual stringettes". Of course Monty Python was making fun of this tendency and 50 years later we just accept it as a part of life.)

[14] Which is now called "HSN". Apparently, we are too busy to pronounce those two extra syllables. Modern life may be difficult, but I don't think the energy I save from not pronouncing those two syllables is going to give me enough energy to overcome it.

[15] Okay, "HSN".

2.3.1 Where does the Unix Principle actually apply in real life?

Kitchen knives — I know that technically, you can use about any knife to cut a tomato or chop an onion. No debate there. But certain types of knives are just better at some things than other types of knives are. Slicing bread is a perfect example. You can slice a loaf of bread with any knife, but a long, fairly rigid, serrated blade works best.

Most people have too many knives, myself included. (Again, this is due to marketing and FOMO—fear of missing out.) But really, you only have three or four basic tasks that you use a knife for:

- Slicing bread
- Cutting meats
- Peeling fruits and vegetables
- Cutting fruits and vegetables
- Cutting cheese[16]

Sure, there are always those oddball tasks that you have to do once a year or less, like using a butcher's cleaver to torque open a pumpkin, but lets talk about those tasks you do on a weekly basis. So let's talk about the knives I have that will accomplish those tasks:

- **Slicing bread** — The afore-mentioned serrated knife with a long, fairly rigid blade.
- **Cutting meats** — If I am cutting meat from the bone, I'll use a boning knife, which has a slightly curved blade that narrows at the tip. Kept sharp, it will also do a great job of cutting up meat for stir-fries, or trimming the fat off the edges of pork chops.
- **Peeling fruits and vegetables** — I don't use a knife for this, because a T-handle peeler is much more efficient, easier on the wrist, and safer.
- **Cutting fruits and vegetables** — Most fruits and vegetables tend to be firm enough that any sharp knife will do. I have a long chef's knife with an 8" blade which is great when I'm chopping a lot of something. (I do wish I had one with a 10" blade, because that would be even more efficient.) I also have a small santoku knife with a thin blade which I prefer when I need to slice something into thin slices,

[16] Yes, I know I said "three or four" and then immediately listed five. Bear with me.

such as cucumbers. It also does a great job of cutting meat into tiny pieces, especially if you put the meat in the freezer for 20-30 minutes first.

- **Cutting cheese** — I have a knife with a 4" blade with holes in it. The idea is that holes prevent the cheese slices from sticking to the cheese. I had an expensive ($15!) cheese knife that did an okay job; it also had a two-prong fork on the end to pick up the slice of cheese with. My current one has a shorter, thicker blade that does an excellent job. I paid $4 for it at Menard's. It simply outperformed the more expensive knife. Sometimes less *is* actually more.

I know that someone out there is itching to point out that tomatoes aren't potatoes and potatoes aren't carrots, *ad infinitum*, and thus the Unix Principle doesn't apply. Well, there are a lot of different pdfs out there, as well, and if I'm skilled with a command line application (such as `pdftk`) it doesn't really matter which pdf I'm dealing with. The job of a knife is to cut. If you keep the knife sharp[17] and *learn how to use it properly*, you'll be a lot more efficient in the kitchen. Notice the emphasis on learning how to use a tool properly. You can learn a lot just by reading the manual. (Go to your terminal and type in `man knife`.)

Breadmakers — I'm going out on a limb here, because a lot of people will be happy to point out that there are all sorts of breadmakers will all sorts of settings. Relax. Breadmakers are designed to do one thing: turn flour, water, yeast, and salt into dough, and then turn that dough into bread. All those settings are just options.

I've resisted buying a breadmaker for years, because I actually don't want a device in my kitchen that only does one thing, and I've always known how to make bread from scratch. But as I get older, I don't always have the time or patience to make homemade bread (it can be a messy process), and a breadmaker is ideal. It does one thing, and it does it really well. (Hint: bread machine yeast is your friend.)

Air fryers — Everything I said for breadmakers also applies to air fryers. They have one job: cook food fast and make it crispy. They work really well on certain items (tater tots!) and not so well on others. I have two analog air fryers, and all you really get to choose is time and temperature. I know

[17] Sharpening a knife is a skill in and of itself. If you watch someone that's good at it, you'll realize that it's also an art form. You can get the skill with a bit of practice; you can learn the art only with long experience. That is really the only way to learn this; there are no shortcuts.

that fancy digital ones have *programs*, but really, they are just different time and temperature combinations, which means less thinking for you. But I like to think (and I like to experiment) so I am perfectly happy with my little analog air fryers.

2.3.2 Where else does the Unix Principle *not* apply that it probably should in real life?

Cars (i.e., portable entertainment centers) — "It's not much, but it gets me where I'm going."

This is how every person who drives a jalopy describes their car. But they are missing a much bigger point. They really should describe their car like this:

"It's not much, but it gets me where I'm going, and more importantly, *it gets me back where I started*."

That is really the only purpose that a car has: to get you from point A to point B and back to point A while doing a reasonable job of protecting you from the elements.

My first car was a 1980 Ford Escort with two doors, a hatchback, an AM radio,[18] and a four-speed manual transmission. It got me where I was going and back again, and it did it in a very economical manner. There was never anything on the AM radio, and FM reception was spotty, so the only entertainment I had was what was out the window, whatever discussion I had with passengers, and my own mind. I would often take long rides in the country on the weekend in it, and since it did not have any reliable way to entertain me, I actually had to *notice* my surroundings. This was the pre-digital age, so there was no mobile phone in my pocket to stop and take pictures with.[19] If I wanted pictures, I had to plan ahead and buy film for my 35mm camera.

Out of all the cars I've ever owned, that is the one with the second fondest memories.[20]

[18] Although the old couple who had owned it installed some excellent speakers and an FM converter, which was a thing back in the day.

[19] Or the ultimate monument to vanity, the selfie.

[20] I could talk about my Chevrolet Corsica, which was the car I owned with the most happy (i.e., quantity), and the happiest (i.e., quality) memories, but that's for another zine.

Nowadays, the purpose of a car is to get you from point A to point B and not allow you to become bored for even a millisecond. Heaven forbid you should get bored on your morning commute. I don't remember ever becoming bored while driving that old car, even though I'm sure I did. But I had a brain that was trained to entertain itself, so such moments were rare and short-lived enough that I don't recall them ever occurring.

Modern cars include satellite radio, seat warmers, DVD players, bluetooth connectivity (we used to settle for 8-track and cassette features long ago, then switched it up to cd players, but now everybody is just streaming their music), GPS navigation,[21] and a bunch of other stuff that has nothing at all to do with getting us where we are going and everything to do with preventing us from getting bored.

Perhaps the best example (or most egregious example, depending on your viewpoint) of this new philosophy was a commercial a few years back for a ~~SAV (suburban assault vehicle)~~ SUV (sport utility vehicle) which featured a young family driving through what appeared to be a wilderness area of the southwest United States. The landscape was simply stunning, and of course, the kids were in the back, watching a movie on a dvd player. I don't recall what the parents in the front seat were doing, but all I can remember is that they were driving through some of the most beautiful landscape this small planet has to offer, and rather than observing that and being amazed by it, the kids are in their own world in the back seat watching a movie they could watch anywhere, and the parents are in their own world in the front seat, flipping through the channels on satellite radio. They could actually make this a wonderful family experience, but no. Why should they inconvenience themselves?

And I know, someone will point out that long car trips are hard on kids, that they don't always find the landscape as beautiful as the adults do. This is all true. But that's no reason to abandon your parental duties. If you can pack a bunch of dvds, you can also encourage your kids to pack up some things of their own choice that they can use to keep themselves entertained. Just because your vehicle enables you to evade your duties as a parent doesn't mean that you should evade your duties as a parent.

Sadly, the irony of owning and being responsible for a car is lost on most Americans. What is your primary reason for owning a car? So you can get to work. And why do you work? So that you can pay rent and

[21] I have to admit that this is useful, and in the age of climate change, I'm all for anything that reduces the number of wrong turns you can make. But I find it easier to go online and plan this out *before* I get behind the wheel of my car. It's probably safer, too.

make your car payment. And why do you own a car, again? So you can drive to work and spend most of your time away from the place that you are spending so much time at work to be able to rent.

There is a part of this equation that makes no sense and could—and definitely should—be deleted. But we, as a society, can't figure that out.

Microwave ovens — Microwave ovens have a lot of buttons because people apparently like to press buttons. (Actually, people like the illusion of choice. Only some of us like to feel like we are piloting the starship *Enterprise*.) But really, most people only use the microwave to do two things:

1. Heat up leftovers.
2. Cook frozen food.

When I was a kid, we were given a microwave oven as a gift. It included a cookbook that had recipes where you could basically make anything in the microwave oven, and at a fraction of the time. Pies, Sunday dinners, fried rice — you name it, you could make it in the microwave oven.

Alas, I have no idea how a beef roast cooked in the microwave oven tastes. Nor do I want to know.

You'll notice that I left "thawing out frozen food" off that list. Have you ever tried to defrost anything in the microwave oven? It's an utter failure, with half of it still being frozen, and the other half being mostly thawed with overcooked inedible bits at the edges, with an odd liquid slowly coagulating on the plate below. No, thank you. The best way to thaw out frozen foods is to plan ahead and throw them in the refrigerator the night before. But in a society which does not encourage us to think—indeed, we are often discouraged from thinking—we are taught to think that defrosting food in a microwave is a good alternative.

It is not.

But we are enthralled with the *illusion* of choice. Most people will not buy a microwave oven with only one or two buttons, even though in reality, that is all you need: one control for how long and another control for how high. My current microwave has only three buttons that I use on a regular basis: 1 minute cook, 2 minute cook, and add 30 seconds. It also has a 3 minute cook, a 5 minute cook, 'Popcorn,' 'Beverage,' 'Potato,' 'Reheat,' 'Delay Start' (why, praytell, are you delaying the start in a device whose entire point is 'right here, right now'?), 'Defrost,' 'Timer,' 'Reminder,' and a host of other buttons for setting the clock, adjusting whether it's AM or PM, etc, in addition to 'Start' (highly useful if you're

not using the 1 minute cook button) and 'Cancel' which I don't use because I just run out the clock. If you pull out food before the timer runs out, the oven keeps giving you a message on the screen that you still have time on the clock. Any well-designed microwave oven should just time out that message after five minutes.

Again, we are enthralled with the illusion of choice, and actually devote time and resources to it, even though they could probably be better spent elsewhere. Case in point: Every microwave oven has a 'Popcorn' button, but every packet of microwave popcorn has an instruction telling you explicitly *not* to use the 'Popcorn' button. Something does not align here.

2.3.3 Where does the Unix Principle not apply in real life and this is actually a good thing?

Instant Pots — and to a lesser degree, multicookers.

If you don't know much about cooking, you can take the point of view that these things actually *do* follow the Unix Principle, because all they're supposed to do is cook food, and they do that very well. But once you learn more about cooking and learn how complex it is, you'll find that there's cooking and then there's cooking. It's like dancing. Just because you know how to do the Virginia Reel does not mean you're ready to star in *The Nutcracker*. They are completely different modes of dancing.

I originally bought a multicooker because the lid to my Crockpot™ cracked, and I needed to replace it. But Costco had a multicooker on sale, and I thought I would try it because if I didn't like it I could just return it.

As a slow cooker, it does a fair job, although not quite as well as my original crock pot.[22] But it also does a fantastic job cooking rice (both white and brown) and steaming meats and vegetables, neither of which I could do in the crock pot. In fact, I also had a steamer which I quickly gave away. I never owned a rice cooker, but once I discovered how easy it was to make rice in the multicooker, I realized that I would never need one. It also has a "soup" function and an "oatmeal" function, but I have yet to try those.

The rice cooker I never owned and the steamer I did own both follow the Unix Principle in that they did one thing and did it well. But the

[22] Dropping to lower case and two words here, because the official name is Crockpot™ and yes, it is trademarked.

multicooker? It follows the same path as my Instant Pot®[23] does by being able to cook in ways that previously required multiple devices.

Again, Costco came through with a $65 sale on Instant Pots. Besides pressure cooking I can steam vegetables, make rice, make boiled eggs (good-bye countertop boiled egg maker!), and make yogurt. It also has a sous-vide function, a soup /broth function, a bean/chili function, a sauté function, and a meat/stew function. In addition, it also has functions for porridge and multigrain, which I have yet to use. That $65 Instant Pot has replaced appliances I don't even have.

These are violations of the Unix Principle that actually work well and that I can live with. The key difference is not that these things have replaced *functions*, but rather, that they have replaced entire *devices* that used to execute those functions.

Well, I've rambled a bit here. I'm sure I'll remember more things to like about Linux after I put this issue to bed. And I'll do a bit of research, as well. But one of my favorite reasons is this:

```
$ cowsay "Linux Rocks"

< Linux Rocks >
 -------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

[23] Yes, that's a registered tradmark also, although most people just pronounce it (and often spell it) as "instapot".

Chapter 3

What Are All Those Folders in the Linux Root?

If you're using a Linux distro with a GUI (Ubuntu, Puppy OS, Mint, etc.) you land right in your Home folder whenever you click on "Files." But if you've ever gone all the way into the root of your computer (the Windows equivalent would be `C:\`) you'll see a lot of folders[24] there with mysterious three-letter names. Let's take a look at the them and what they contain.

(For more information about this, consult the Linux Foundation *Filesystem Hierarchy Standard*, which is found at `https://refspecs.linuxfoundation.org/`. You're probably going to want the pdf version of this, which is at `https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf`. It really is amazing how much you can learn just by reading the specs and manuals. Scotty was right.)

The essential point behind this directory structure is to segregate different file types. To quote the FHS:

> It is possible to define two independent distinctions among files: shareable vs. unshareable and variable vs. static. In general, files that differ in either of these respects should be located in different directories. This makes it easy to store files with different usage characteristics on different filesystems.

[24] Technically, these are *directories*, but let's not be pedantic. In a GUI, the icon usually looks like a folder.

The FHS goes on to clarify these terms:

> "Shareable" files are those that can be stored on one host and used on others. "Unshareable" files are those that are not shareable. For example, the files in user home directories are shareable whereas device lock files are not.
>
> "Static" files include binaries, libraries, documentation files and other files that do not change without system administrator intervention. "Variable" files are files that are not static.

Now, if that excites you (and it does me), I urge you to read the FHS, especially the "Rationale" section of this description, which talks about the historical use of `/usr` and `/etc` and why `/var` was created. (I admit, I'm a bit of a junkie for early history, whether it's mythology and folk tales, or computer systems. I love learning about the early days.)

(I also love the fact that the spec goes out of its way to point out that "unshareable" files are simply files that can't be shared. It's kind of like saying a "non-citrus" fruit is any fruit that is not citrus. You would think stuff like this would be obvious, but it's not always.)

3.1 bin

This directory contains essential command binaries that need to be available for all users. Many of these include binaries that bring up the system or repair it. Your basic binaries like `cat`, `ls`, and `mv` live here.

What exactly *is* a binary? The technical definition is that it's a file that contains compiled source or machine code. They are also called "executables" because they can be run (i.e., "executed") on the computer. Programs, if you will.

3.2 boot

Boot loader files. It's complicated—kernels, and so forth. It is also four letters instead of three. I don't really understand it, but I'll look into it. I believe this has to do with what happens when you start up your machine, so this is prompting me to create an issue #2 to explore this.

3.3 dev

Device files, such as `/dev/disk0/`, `dev/sda1`, etc. Also the home of the wonderful `/dev/null`.

3.4 etc

System-wide static configuration files. It is **not** allowed to contain binaries. Some examples include:

/etc/opt — Configuration files for add-on packages stored in `opt`.

/etc/sgml — Configuration files, such as catalogs, for software that processes SGML.

/etc/X11 — Configuration files for the X Window System, version 11.

/etc/xml — Configuration files, such as catalogs, for software that processes XML.

3.5 home

Users' home directories. If you have multiple users, you will see a directory in here for each user, named after their login name. Their personal settings are stored as invisible files in their home directory.

3.6 lib

This directory contains libraries that are essential for the binaries in `/bin` and `/sbin`. Unlike bin, it only contains system binaries which require root privilege.

lib32 and **lib64** — We are still in a transition between 32-bit and 64-bit systems.[25] These two libraries clarify which register size to use. A 64-bit system may have compatibility for 32-bit binaries.

lib/modules — This directory stores kernel modules (pieces of code that can be loaded and unloaded into the kernel upon demand, adding functionality to the kernel without needing to reboot the system[26]), with each

[25] How is this even possible? SMDH.
[26] Handy, that.

installed kernel having its own directory. Their filenames are identifiable as `ld*` or `lib*.so.*`. The Linux kernel typically loads kernel modules from the appropriate kernel version directory.

To show which kernel modules are currently loaded, run

```
$ lsmod
```

To show information about a module:

```
$ modinfo module_name
```

To list the options that are set for a loaded module:

```
$ systool -v -m module_name
```

To display the configuration of a particular module:

```
$ modprobe -c | grep module_name
```

To list the dependencies of a module:

```
$ modprobe --show-depends module_name
```

3.7 media

Mount points for removable media. Using a jump drive or SD card? This is where they are.

3.8 mnt

Temporarily mounted filesystems.

3.9 opt

Add-on application software packages. When some applications are installed directly from source, they will place themselves in `opt`.

3.10 proc

A virtual filesystem providing process and kernel information as files. In general, the system automatically generates and populates these files.

3.11 root

The home directory for the root user. I'm not entirely sure what this means. On my computer (Ubuntu 20.04) it's an empty directory, owned by root/root, but even root doesn't have access rights to it. This requires more investigation.

3.12 run

Run-time variable data. That is, information about the running system since the last boot, such as currently logged-in users and running daemons.

3.13 sbin

Essential system binaries that configure the operating system, such as `fsck` and `init`. Unlike `bin` it only contains system binaries which require root privileges to run.

3.14 srv

Site-specific data served by this system, such as data and scripts for web servers, and repositories for version control systems.

3.15 sys

This directory contains information about devices, drivers, and some kernel features.

3.16 tmp

Temporary files, natch. (These may also appear in `/var/tmp`.)

3.17 usr

This directory is a secondary hierarchy for read-only user data, and contains the majority of multi-user utilities and applications. This should be shareable and read-only.

3.18 var

This directory contains files who content is expected to continually change during normal operation of the system, such as logs, spool files, and temporary email files.

Chapter 4

A Scanner Darkly, but with a workflow

I suppose I should have been an archivist. I am always trying to preserve the written word in digital form.

And this makes sense. It's easier to share a digital file of something than to share the thing itself, because as my experience with sharing books highlights, you rarely get them back. Also, the further you spread something, the more like it is to be preserved. *Preservation through dissemination.*

So I scan a lot of things. Because this can be a messy, complicated process, I've developed workflows around this. (I am big into workflows, because once you have one down, it's easier to anticipate and deal with interruptions or disruptions, unless you run into a mule.[27]) So here is my workflow for scanning things.

My hardware is a Brother MFC-J805DW printer/scanner/fax machine.[28] And this is where we run into problems, because while Brother does make Linux drivers for this machine, the printer driver works great and the

[27] If you've read Asimov's *Foundation* series, you'll recognize that reference.

[28] One day, we will eventually give up faxing, which is archaic at this point. I don't know if we'll just start calling these machines "printer/scanners" or if we'll continue to call them "multi-function machines" because they still can make copies. Futurists tend not to care about the details. (In reality, these will all be obsolete in the new digital order, when the oceans have risen and all the paper underwater has decomposed. I'm not a futurist, so I'm interested in the details.)

scanner driver does not. If I install it, it works fine for three or four scans and then it starts to hang. I can uninstall it, reinstall it, and get a few more good scans out of it before everything goes pear-shaped again. I could live with this if I only did the occasional scan, but I scan on a regular basis.

Alas, this is the one case where I have had to rely upon commercial software: VueScan. The company which produces it, Hamrick Software, creates their own drivers, updates it often, and responds to issues incredibly quickly. It costs me $100 a year, but this is money that I am happy to pay. (And if there were an open-source version of this software, I would be happy to pay that $100 to it, as well.[29])

My Brother scanner does *not* have a duplex scanner. Since books are printed on both sides of a sheet of paper, this presents a problem. But before we dig into things, let's get some terminology out of the way.

A **sheet** of paper (also called a **leaf**) has two sides, which are called **pages**. If you open a book, you'll notice the pages on the right side have odd page numbers. This is the **front side** of the pages, also called the **obverse** side. The pages on the left side have even page numbers. This is the **back side** of the pages, also called the **reverse** size.

My basic workflow works like this:

1. Separate the document to be scanned into groups of ten sheets.
2. Scan the front side of a group.
3. Scan the back side of a group.
4. Interleave those two scans, so that the front sides and back sides are in order in a single pdf.
5. Repeat until all sheets have been scanned.
6. Concatenate all the two sided scans from step 4 into a single document.

It helps to be have a naming scheme for your files, because you're going to end up deleting a lot of them. So let's run through this again, and include file names.

1. Scan the front side of the first group and name that file `001a.pdf`.
2. Scan the back side of the first group and name that file `001b.pdf`.
3. Interleave the two scans, and name the resultant file `001.pdf`.
4. Delete all files that have `a` or `b` in the file name.

[29] It also has an auto-deskew function, which is handy when the paper I am scanning is narrower than the minimum width my document feeder can handle.

5. Concatenate all the remaining files into a new filename that *doesn't* have numbers in its filename.
6. Delete all files except the one we created in the previous step.

All of the scanning is done through VueScan. Afterward, the magic happens with `pdftk`. (See `www.pdflabs.com/tools/pdftk-the-pdf-toolkit/`.)

Okay, how do we do all of this stuff with pdftk?[30]

We can interleave two scans with the pdftk command `shuffle`. The basic command looks like this:

```
$ pdftk A=001a.pdf B=001b.pdf shuffle A B output 001.pdf
```

What's happening here? pdftk allows us to assign *handles* to files whenever we want to use just parts of those files, whether it's a single page or a range of pages. In this case, we are assigning the file 001a.pdf (which contains the front, odd-numbered pages) to the handle 'A' and the file 001b.pdf (which contains the reverse, even-numbered pages) to the handle 'B'.

We are then telling pdftk to interleave the two files via the `shuffle` command. So if A contains the pages 1 3 5 7 9 and B contains the pages 2 4 6 8 10, the output pdf (001.pdf) will contain the pages 1 2 3 4 5 6 7 8 9 10. Pretty nifty, huh?

However, when you flip the group of sheets over and run them through your scanner, the reverse sides will be scanned in reverse order, because the highest numbered page will be the first to be scanned. So 001b.pdf will actually contain the pages 10 8 6 4 2. This is a problem.

Fortunately, pdftk has a way around this. Take a look at this command:

```
$ pdftk A=001a.pdf B=001b.pdf shuffle A Bend-1 output 001.pdf
```

The `Bend-1` means "input file B, but start at the end, and work backward to page 1".

In the end, I end up with a directory full of files based on this pattern: 001a.pdf, 001b.pdf, 001.pdf. Once I've verified that all the shuffled files are correct and complete (if 001a has 12 pages, then 001b should also have 12 pages, and 001 should have 24; if not, something got either missed or repeated), then I can clean house with this command:

[30] I won't bore you with how to install it, since there are multiple ways to do that. You can easily find this information online.

```
$ rm *a.pdf *b.pdf
```

We now have all our two-sided files (001.pdf, 002.pdf, etc.) and none of the one-sided files. Now we can combine these into a single file:

```
$ pdftk *.pdf cat output book.pdf
```

And to get rid of all the other files:

```
$ rm 0*.pdf
```

Our revels now have ended. Go forth and scan.

Chapter 5

Is This Really a Hack? Or Is It Just a Tip?

The word "hacker" has a lot of definitions, and if you google it, you'll find a lot of scary ones on the websites of companies that want you to be scared of hackers and then spend hundreds of dollars on their security products, some of which may actually protect you against actual threats, and some of which may provide protection against an imagined threat.

(And yes, there are bad people out there who use their advanced technical knowledge to attain access to systems that they shouldn't have in order to obtain information they're not supposed to have. I'm not talking about those people, who technically should be called "crackers," rather than "hackers," a lá "safe crackers".)

Rather, I'm talking about the older meaning of the term "hacker" which is somebody who enjoys the intellectual challenge of pushing software (and often hardware) beyond what it is meant to do in order to achieve interesting and clever outcomes. In order to do so, of course, they have to know the systems they are working with fairly well. In fact, the definition of "hack" that I like best is "an appropriate application of ingenuity."[31]

Of course, this term originally referred to computer technology, but now I'm finding that people are using it everywhere, even in places where it doesn't belong. (I'm looking at you, the writers and editors of apparently every food magazine and website ever.)

[31] See `http://www.catb.org/~esr/jargon/html/meaning-of-hack.html`.

So let's look at some things that have been called "hacks" but may or may not be actual hacks. These all came from cooking websites that were at the top of a google search for "cooking hacks."

1. **How to cut up a mango** — Not a hack. In fact, I'd argue that this is just basic knowledge. Cut up one ripe mango the wrong way and you'll be googling the right way to do that pretty darn quick.
2. **Use an ice cube tray to make sushi (in lieu of using a bamboo sushi roller)** — While you are using a common device in an uncommon way, I don't think this rises to the level of a hack. There are a lot of ways to put raw fish and rice together.[32] And honestly, just learning to roll sushi would be easier, considering how difficult it is to get ice cubes out of those trays sometime.
3. **Make hot chocolate cocoa bombs** — Not a hack. A technique, to be certain, but not a hack.
4. **Punch holes in your sausage with a toothpick before cooking to keep them from exploding** — A useful tip (especially in the air fryer), but definitely not a hack.
5. **Use a spiralizer for perfect baked curly fries** — Really? Next we'll have "Use a hammer to pound nails into wood" or "Use a pencil to make marks on paper." Using a tool for what it's meant to be used for is not a hack. The unique thing here is "baked" and that has nothing to do with a spiralizer.
6. **Peel ginger with a spoon** — This is most definitely not what you're supposed to use a spoon for. But this works great. I'll call this one a hack, but just barely. If I used ginger all the time, I'd probably just call this a technique.
7. **Microwave lemons and limes to get more juice out of them** — A tip, not a hack.
8. **Freeze cheese to make it easier to grate** — I do this all the time, and it's a great technique. The trick is leave the cheese in the freezer for just the right amount of time. Not a hack.
9. **Refrigerate onions before chopping them so they don't make you cry** — Chilling the onions makes the volatile sulfur compounds in them less volatile, so this works, provided you chop fast. It's not a hack, but it is a good technique.

[32] Also, be sure to sanitize the hell of that ice cube tray before you use it to make ice again.

10. **Wear swim goggles while chopping onions so they don't make you cry** — I'll call this one a hack. Goggles normally protect your eyes from liquid water. Using them to protect your eyes from volatile sulfur compounds is an appropriate application of ingenuity, since I doubt most home cooks have lab quality eye goggles.
11. **Use parchment paper in place of muffin liners** — The purpose of parchment paper is to get between your food and whatever it's being cooked on or in. If you run out of muffin liners, you can just use squares of parchment paper. This is just using parchment paper for what it's meant to be used for, albeit in an atypical way because we have another product that we use in this instance. A great technique to use on a regular basis; a great tip if you've never heard of it before. But not a hack. (I actually think the corners sticking it up would make it easier to get the muffins out of the pan.)
12. **Flavor your pasta water with a chicken stock cube** — Not a hack. Anyone who grew up eating instant ramen on a regular basis figured this one out by the time they hit double digits.
13. **Use a potato masher to break up ground meat** — While technically a potato masher is meant to be used on potatoes,[33] its ultimate purpose is to mash big things into little things.[34] This is like using a screwdriver to pry up a lid of paint. It's not a hack because it's so obvious.
14. **Collect all your vegetable scraps in a freezer bag and use them to make stock** — Not a hack. Just frugality in action.
15. **Buy the biggest cutting board you can find** — While this is great advice, it's not a hack.

So, 15 "cooking hacks" and only two of them are actual hacks, and one of them is fairly questionable. I'm pretty much calling it a hack because I'm trying to be generous. The thing that strikes me is not so much that these aren't actually hacks, but the difference between a "tip" and a "technique." Maybe I'm just tired, but it seems like it's a technique if you use it on a regular basis and a tip if you're in a jam and someone tells you about it. But that's an argument for another day in another zine.

[33] It's right there in the name.
[34] It's right there in the name!

Chapter 6

Coda

6.1 What I Learned About LaTeX While Creating This Issue

I'm still a relative newbie to LaTeX, so there's always something to learn. Here's a running list of what I've learned so far:

1. You might think you want the **book** document class, but you probably will find the **report** class just as handy.
2. You want links?[35] Use the **hyperref** package.
3. The **kpfonts** package has beautiful fonts. You're soaking in them now.
4. Footnotes are easy! (Seriously, footnotes in LaTeX have got to be the easiest footnotes I've ever managed.)
5. Use the **fancyhdr** package to get more granular control over your headers and footers.
6. You can use the **geometry** package to make a document have a paper size of half letter (i.e., 5.5 inches by 8.5 inches).

[35] Yeah, I know these are irrelevant in a paper document.

7. You can make your top margin larger by using `\addtolength {\topmargin}{0.5in}` but there is not a similar parameter for the bottom margin. Instead, you need to make the text box shorter by using `\addtolength{\textheight}{-1in}`.

8. Want to show code blocks? Use the `\begin{verbatim} code block \end{verbatim}` construction. (Line breaks are up to you.)

9. Want to show inline code without executing it? Use `verb` followed by two pipes. Place your code between the pipes. (I had to use two of those in #7, because that code just went right off the edge of the page when I only used one.) You can also just use `\texttt{}` if you want something in a mono-spaced font, but it will not display raw LaTeX code.

10. Need a little space between elements? Just insert `\,` (that is, a backslash followed by a comma). (This is actually a non-breaking space, so use it judiciously.)

11. The above item is because whenever I invoked `\LaTeX`, the space between it and the next item would disappear. Turns out that you should always invoke it with an empty argument (i.e., `\LaTeX{}`) because without the empty argument, LaTeX is simply looking for the end of the command (i.e., the space) and then moves on. The empty argument tells it that the command is over and to move on. (For more information, see this: `https://tex.stackexchange.com/questions/31091/space-after-latex-commands`.

12. Footnotes reset back to the number one with each chapter. To prevent that, add `\counterwithout{foootnote}{chapter}` to the preamble.

Like I said, I'm still a newb and I may be completely wrong or off base on some of these things, in which case, I'll make a note of that in a future issue [36]

If you are interested, there is a link in the Impressum to the git repo for this publication where you can check out the source code.

[36] Always assuming that there *will* be another issue.

6.2 Why LaTeX?

There are lots of reasons why I wanted to use LaTeXto create this zine. I've been learning the language on my own lately, and I really just wanted to set myself a new challenge. It's always good to have challenges. If you poke around the website where you can find this zine, you'll find some other projects that are also typeset in LaTeX.

As for why I wanted to learn LaTeX in the first place—well, that's complicated. Any kid that is interested in computers and science is probably also into math, and I've certainly never been an exception. I've written up a lot of math stuff, first in Microsoft Word[37] and later in LibreOffice, and while it's not the most terrible experience, it's not exactly enjoyable, because the programs themselves are not centered around math formulas.

I knew in the back of my mind that there was this great typesetting thing specifically for math and scientific papers, but it took me a while to find it. (This was back in the days before the internet.) But I could recognize just about any paper typeset with it.

Is LaTeX the ideal thing to create zines with? Not necessarily. It does give the finished product a very polished look, and although I love the handmade look (i.e., "clip and copy") of a lot of zines, I don't have the talent (or the patience) required to pull that look off. The important thing for me is to be able to create something that is easy on the eyes. (And I've seen a lot of handmade zines that *aren't* easy on the eyes. Like I said, this takes talent.)

I've created zines in the past, and I've previously used Adobe InDesign (back in my Macintosh days) and LibreOffice Writer (more recently). They are WYSIWYG[38] programs and it's fairly easy to get what you want (more or less).

(They can be a pain in other ways, however. InDesign used a lot of resources on the little Mac Mini I was using and system crashes, while not frequent, were also not uncommon and always poorly timed. LibreOffice has certain interface quirks that I just find frustrating, such as using sections to have different footers or headers. You can't have everything.)

However, they aren't very amenable to version control, because they don't generate text files, but proprietary files. (I'm not sure about InDesign files, but .odt files are just a collection of `.xml` files and a few others. Change the `.odt` to `.zip` to see them.) While you can track these files in

[37] Insert the "horses neighing nervously" sound from *Young Frankenstein* here.

[38] What You See Is What You Get

with version control software such as `git`, you can't see the differences between those files by running a simple `git diff`. You have to download the files in question, make sure they have different names so they don't overwrite each other, and check for differences manually.[39]

With LaTeX however, your `.tex` file really is just a text file, and so if you use something like `git` to track changes, you can run a `git diff` and see changes from one commit to the other pretty easily. This is nice when you are writing, because if something isn't working, you can just delete it, and if you decide you can use it after all, you can just go back to the repo and get it.

With LaTeX, it can be difficult to get something to appear exactly where you want it to. Placing objects is actually easiest in InDesign. The main advantage of LaTeX is that it makes it *extremely* easy to make things consistent. I suppose this is something that I will get better at as I gain more experience with it and learn about some more packages.

Also, there is no easy way to get a word-count from a LaTeX document, nor is there a spell-check. Considering that LaTeX was originally constructed for use in academic contexts, I find this strangely lacking. You can get around this by converting it to another file form and counting the words and running a spell check there. I used

```
$ pandoc 001.tex -o 001.odt
```

to convert this `.tex` document to a LibreOffice document and counted the words there. (It couldn't find the two images I include in this document, but that's okay.) Before I added the last three sentences, I was at 6,515 words. I didn't bother to do a spell-check.

I'm not where I want to be with LaTeX yet (and I definitely won't be for a while—I've got bills to pay) but perhaps the nicest thing about LaTeX is that while there are a lot of packages available, if you can't find one to do what you want to do, you can always create your own. It will be a while before I get to that point because first I need to find something I want to do in LaTeX that isn't covered by an existing package, but I someday might. Remember, with the level of control you get with Linux, you also get opportunity. And it's always good to have a challenge to look forward to.

[39] As a result, to do version control, I used a version number in the file name and simply did a "save as" every time I opened the file for editing, incrementing the version number as I did so.

6.3 What's Next?

Issue #002, maybe? I know I learned a lot about the file structure (chapter 3) and there's a lot more research and learning to do there. I'd also like to delve more into the history of Linux itself, because Linus Torvalds was developing it just as I was finishing my bachelor's degree. (I was on the seven-year plan for that.)

Also, I've dusted off my bass guitar and keyboard and I plan to start playing music again. I need to learn a lot more about music theory, and I'm thinking I may write up my notes in LaTeX, just so I can learn how to typeset music with it.

Dedication

For Travis, who kept things simple.
For Ryan, who made things complex.

Such is the nature of existence.