

the codex

Life with Linux — A Zine

Typeset in L^AT_EX


Issue #003

Kenneth John Odle

July 8, 2023

Impressum

All contents ©2023 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^AT_EX 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/>).

The image of Linus Torvalds on the front cover is courtesy JericoDelayah from the Wikimedia Commons. The image is from https://commons.wikimedia.org/wiki/File:4_RETAT_04_Linus_Torvalds.jpg where you can also find a link to the Creative Commons CC BY-SA 3.0 license there, as well.

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.

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 picture of a VT100 terminal is courtesy of Jason Scott. It was published at https://en.wikipedia.org/wiki/VT100#/media/File:DEC_VT100_terminal.jpg where you can also find the Creative Commons 2.0 license it was licensed under. (By Jason Scott - Flickr: IMG_9976, CC BY 2.0, <https://commons.wikimedia.org/w/index.php?curid=29457452>)

In This Issue...

1	The Final Salad Days	5
1.1	College, 1986	5
1.2	Teaching Computers	7
2	Is This Really a Hack?	
	(Or is it something even worse?)	10
3	Music (♪) in L^AT_EX	15
3.1	Standard Notation	15
3.1.1	wasysym	15
3.1.2	harmony	16
3.1.3	musixtex	16
3.1.4	ABC	17
3.2	Tablature	17
3.2.1	guitartabs	18
3.2.2	songs	18
3.2.3	musixtex	19
3.3	Lilypond	19
3.4	Summary	19
4	An Introduction to L^AT_EX	20
4.1	Files	21
4.2	Paragraphs and White Space	21
4.3	File Structure	21
4.3.1	The Preamble	22
4.3.2	The document Environment	22
4.3.3	Publication Structure	22
4.3.4	Environments	23
4.3.5	Math in L ^A T _E X	25

- 5 What’s to Like About Linux** **27**
- 5.1 Open Source 27
 - 5.1.1 Open Source Software is Available for Free 27
 - 5.1.2 You Can Alter the Source Code 29
- 5.2 Containerization 31

- 6 Ubuntu 22.04** **33**
(and Ubuntu 24.04)

- 7 Coda** **35**
- 7.1 What I Learned About L^AT_EX While Creating This Issue 35
 - 7.1.1 Image Sizes 35
 - 7.1.2 Installing L^AT_EX Packages on Ubuntu 36
 - 7.1.3 Miscellaneous Things 37
- 7.2 What I learned About L^AT_EX While Creating Something Else . . . 38
 - 7.2.1 Removing Page Numbers from “Part” Pages 38
 - 7.2.2 Adding Additional Text to “Part” Pages 39
 - 7.2.3 Adjusting the Line Spacing 39
 - 7.2.4 Include Page Numbers on “Chapter” Pages 40
 - 7.2.5 A Few Little Things 40

Chapter 1

The Final Salad Days

1.1 College, 1986

I went away to college in the fall of 1986. Personal computers were, as always, very much on my mind, but all I had was a Commodore 128. We were still typing papers on electric typewriters at that point, and the height of that technology was correction paper.¹

We did have computers available to us, however. Our college owned two mainframe systems: a DEC-10 and a VAX 9000. As I remember, you had to go down to the basement of the science building to use them, where they had a room filled with VT100 terminals.

I wish for the life of me that I could remember this experience better. The “computer room” (nobody thought of it as a “lab”—it would be many years before I heard that term applied to it) looked a lot like what one might think: a windowless basement room with concrete block walls, tile floors, fluorescent lights buzzing

¹This was a very convenient alternative to a product called “Liquid Paper” (also called “whiteout”) which was essentially paper-colored paint in a small bottle with a brush in the cap. When you made a mistake, you shook the bottle and brushed a very thin layer over the mistake. You then waited for the whiteout to dry, backed up, and typed the correct letter. The fluid contained a lot of solvent, and as this was the time of the Satanic Panic, parents were warned that their kids could be huffing whiteout. As you used the product, more and more of the solvent evaporated out of the bottle, until it eventually became a gloopy, chalky mess, meaning you could only use half of what was in the bottle—it was not very efficient. With correction paper, you just backed up to the mistake, put a bit of correction paper on top of the paper, typed the mistake again, removed the correction paper, backed up, and typed the correct letter. It was much neater, had no terrible fumes, and you didn’t have Fundies chasing you down the street accusing you of being a devil-worshipping drug addict because you had a bottle of correction fluid in your pocket. (Liquid Paper was invented by Bette Nesmith Graham, who also happened to be the mother of Mike Nesmith of the Monkees. The world is much smaller than we think it is.)

away like hornets overhead, and lots and lots of terminals.²



People didn't really know much about these two mainframe systems, although I remember hearing the few computer people who were around praising the VAX as being far superior to the DEC-10.³ Everything was from the command line. If you saved a file⁴ and wanted to print it, you had to send it to the print queue, and then go to a different room in the building where the line printer was located.

At this point, it was possible that your printout would be ready. But it was also entirely possible that it wouldn't be ready. Everything was printed on unperforated continuous feed paper that was 15 inches wide.⁵ The problem was someone had to be there to tear off your printout after it was completed—the printers had no way of doing this automatically—figure out that it belonged to you, label it with your name, and set it on a shelf for you to pick up later.

And if nobody was there, the printer just kept printing, and someone would have to separate potentially dozens of different print jobs. If your print job was small, the person responsible could miss it and it would end up stuck on the end of someone else's print job. If the printer ran out of paper and nobody was there to replace it, your file just went into the printer memory (or somewhere into the ether if the printer's memory was full) until the paper was refilled. Printing anything was a big investment of time and energy (not to mention hope) and I will gladly take the occasional printer jam over that experience any day.

The only other remarkable thing I remember about that early college experience is that everybody had to take a basic computer course. This was a single, university-wide, 100-level course that all freshmen had to take.

I hate these sorts of things.

I get the point—computers are going to be a thing in everybody's lives, so let's make sure all our future graduates have a solid background in them. The problem is that information technology moves at a pretty rapid pace, and college students have a wide variety of backgrounds and career plans, and as a result, it's difficult to create a course like this that is in any way useful to every single student who takes it.

²There is a pdf of the VT100 manual available at <https://vt100.net/dec/ek-vt100-tm-002.pdf>. It makes for fascinating reading, assuming you are into that sort of thing.

³A bit of internet research confirms this—the VAX line of mainframes was intended as a replacement for the DEC line of mainframes.

⁴I tried for a few weeks to type up some of my notes from class, but quickly realized that this was pointless, as I couldn't take the digital files with me.

⁵See the Wikipedia entry on "continuous stationery" to see what I'm talking about.

And I found that I knew much of the material anyway. It was held in a large lecture hall, and there wasn't really any point for the instructor to hold up a hard drive in her hand and announce "This is a hard drive"—unless you were in the first ten rows, she could have been holding up a brick.⁶ I'm sure that what she said after that sentence was informative, but I'm also sure I already knew it.

Sometime during the third week we had an entire lecture about modems ("modem stands for *modulator-demodulator*" I remember hearing before I dozed off⁷) and I decided to just stop attending class. I showed up for the exams, rushed through them, got a C in the course, and decided to never take another computer course ever again.

I should mention that I went to college with the goal of becoming high school biology teacher. Over time, I realized how valuable this computer course was to me, because it made me realize that no matter who you are or where you are in life, any class should be valuable to you in some way, even if it's not the way the course designers intended. This course is now valuable to me because it taught me what a class should *not* be. (This is a lesson that the modern educational-industrial complex has not, and will never, absorb.)

1.2 Teaching Computers

I graduated from college with a B.S. in Biology and a teaching certificate. I could not find a job teaching biology—I quickly discovered that biology teachers are a dime a dozen⁸ and chemistry teachers are about five bucks a dozen, and also that if you majored in physics with a goal of teaching high school science, you would have your choice of any teaching position you wanted.

Of course at this time (the early 90s), lots of people still wanted to be teachers (unlike now) because teachers still mostly garnered respect from the public, parents, and administrators, and teaching jobs were hard to come by (also unlike now). So I started substitute teaching and doing whatever I could to pay the bills.

I eventually managed to find a job at my old high school teaching what used

⁶This is not hyperbole—drives were a lot bigger in those days.

⁷Which is true, but means absolutely nothing to anyone who just wants to get on the internet. Modems were designed to transmit digital data over analog telephone lines, and they accomplished this by *modulating* a carrier wave to carry digital information before sending it, and *demodulating* a received carrier wave to recreate the original digital information. I don't miss 4,800 bps speeds, but I do miss that modem login sound. Apparently other people do as well, because you can hear it at <https://www.youtube.com/watch?v=gsNaR6FRu00>

⁸This was a huge surprise to me, because all through college, whenever I told someone I was a biology major, they seemed really impressed and said something like "oh gosh, biology—that's really hard." But note—these were non-science people. If you want to study science, but don't want anything too hard, apparently biology is the default.

to be called “night school”⁹ and what is still called “community education”. But names are only labels and are usually irrelevant. My night school students had dropped out of a traditional high school education and were now in search of a GED¹⁰ to help their job prospects. I was there to teach them just enough Earth Science to enable them to pass the science portion of the GED exam.¹¹

More relevant to the purposes of this zine were the community education classes which were your basic “Introduction to Computers” class.

I really wish I could remember how I got this job. I would like to think that I had a very long, very detailed job interview which I simply aced, but I very much doubt it. I don’t remember meeting with an administrator at all. The only person I can remember interacting with was a much pressed-upon administrative assistant named Geri. (I would not have been surprised to discover that she supplemented her income by delivering pizza on the weekends—such was the level of our funding.)

So yeah, an interview in which I was grilled about my knowledge of both computers and pedagogical theory did not happen. What probably happened was that I applied for the position teaching Earth science and submitted a paper resume written on a Commodore 128 and printed out on an Okidata dot-matrix printer, and they saw that dot-matrix printing and thought “This guy knows how to use a computer. We should ask if he can teach our community education computer course as well.” In fact, I’m quite sure that is what happened. Dot-matrix for the win!¹²

I don’t remember much about this “Introduction to Computers” class. It took place in the same room my high school computer class had been in, but all the previous computers were gone and had been replaced by shiny new computers running Windows. As this was around 1992, this would have been some version of Windows 3.

What I do remember the most about this class was that the best way to teach anybody anything about computers was to maintain a completely hands-off policy. That is, if somebody asked me how to do something, rather than grab their mouse and *demonstrate* how to do it, I found that it was better if I stood back, told the student to grab their mouse, and then told them where to point it and where to click. I confess this was partly laziness on my part—you can only *show* someone so many times how to do something as simple as printing a file or turning the computer on before you are completely done with it.

But I discovered a wonderful thing about this: *telling* is very different than

⁹I have no idea what this is now called, or if it is even still a thing.

¹⁰General Equivalency Degree—aka “high school diploma in a box”.

¹¹There is so much that I could say here, but it is completely irrelevant to our current purpose and so belongs to an entirely different zine.

¹²I have no problems with this. Sometimes low expectations work in your favor. Don’t look a gift horse in the mouth.

showing. When I show someone something I tend to use words like “here,” “over here,” etc. But when I *tell* someone something, I have to use much more specific terms like “upper-left hand corner” and “half way down”.

This meant that when I told someone how to print a file I ended up saying something like “move your mouse¹³ to the upper left-hand corner, find the ‘File’ menu and click on it, and then go about half way down until you see the word ‘Print’ and click on it.”

And this worked. My students were not familiar with a “File” menu, but they were familiar with the concepts of “up,” “down,” “right,” and “left”. This led me to realize something that I had not been taught in college—you have to work with students where they are, rather than where you wish they were. You can’t play the “if only” game. (“If only my students knew where the File menu is. . .” Again, there is a lot more to say here, but that’s an entirely different zine.)

I know that the usual dictum is “show, don’t tell”. But what’s really happening here is that by *telling* my students, they were then *showing* themselves, and developing some muscle memory along the way.

Although I thoroughly enjoyed teaching this class, nothing good can last forever. A new Republican governor was elected and he slashed funding for community education and adult education programs. (A less-educated populace is easier to control, I guess.)¹⁴ I taught this course for a year, had a great time, and would gladly teach it again, even with the miserable wages. Hell, I’d do it now as a volunteer. Knowledge should be shared, not hoarded and sold.

¹³i.e., cursor. To someone who is new to computers, the two are one.

¹⁴He also slashed anything that benefits anyone who isn’t wealthy *and* white *and* male. (Mathematically, that would be *wealthy* \wedge *white* \wedge *male*, with emphasis on the *and*.) To this day I would pay good money (i.e., beer money) to buy him a one-way ticket to Planet Fuck-Yourself-Up-The-Ass-With-A-Sharp-Stick.

Chapter 2

Is This Really a Hack? (Or is it something even worse?)

Way back in issue #1, I took a look at cooking “hacks” and wondered if these were really hacks (i.e., “an appropriate application of ingenuity”) or if they were merely being marketed as a hack because in the past few years, hackers and hacker culture have become a trendy thing to talk about, even though the public really has no idea what those words actually mean.

You keep using that word. I do not think it means what you think it means.

—Inigo Montoya, *The Princess Bride*

It really bugs me that the words “hack” and “hacker” have become associated with “cool” and “must have” thanks to marketers. (It also bugs me that a whole class of employees exist whose only purpose is to convince you to buy things that you probably don’t need. But here we are.¹⁵)

So I decided to attempt this quest again, but instead of cooking hacks, I decided to look at gardening hacks, as I know quite a bit about gardening. I went to the Google, typed in “garden hacks” and jumped into the first result that came up.

¹⁵My issue is not that they are here to inform you (if their products were *that* great, an informed decision is all I need to make), but to influence your behavior in ways in which you probably aren’t even aware.

1. **Use newspaper as a weed barrier** — That’s right: just lay some newspaper down on the ground, throw some dirt over it, and go to town planting your garden. This is definitely not a hack, it’s more like a **gimmick** that is actually **really bad advice**. Newspaper will break apart quickly, and is not effective against perennial weeds unless you lay down a really thick layer. Besides, the advice was to put dirt *on top of the newspaper*. What’s to keep wind-blown seeds from just landing and sprouting on *that* dirt? Save your money and just buy some mulch.
2. **Use plastic bottles as mini-greenhouses** — I’ve seen this so many times and its popularity seems to rely on the fact that people somehow think of greenhouses as magical boxes.¹⁶ The point of an actual greenhouse is to let light in. The watering is still up to you. So yeah, you can cut a bottle in half, fill the bottom with soil, plant your seeds, and throw the top on to keep moisture in until the seeds sprout, but it seems to me it would be easier to just plant the seeds in the bottom and make sure to keep them watered. There is nothing magical about a transparent top, and thus this isn’t a hack, but a mere **gimmick**.¹⁷
3. **Punch some holes in the cap of a gallon milk jug and use it as a watering can** — This is definitely a **gimmick**. Why not just leave the cap off and pour water directly out of the jug the way you do milk? Is that not simpler? The *real* hack is to drill a few small holes in the *bottom* of the jug, fill it with water, and set it next to your plants. This is a great way to keep tomatoes and other large plants watered during a hot dry summer without constantly sprinkling them with water.
4. **Place a kitchen sponge in the bottom of a pot to soak up extra water and avoid root rot** — The problem with this **gimmick** is that sponges absorb water and hold onto it until it evaporates (and not a lot of evaporation is going to happen if it’s been buried). If you give your potted plant too much water, the ideal situation is to have something large—stones, for instance—that don’t lock together that will keep the dirt in while letting the excess water out.¹⁸ A sponge will just hold all that extra water, making this **really bad advice** if you tend to overwater.

¹⁶Clarke’s Law applies to greenhouses, apparently.

¹⁷I suppose this is popular because people can then say “hey, look at me, I’m recycling!” but you aren’t recycling, you’re *reusing*. And the end result is a dirty bottle that *can’t* be easily recycled.

¹⁸Or you could just learn how to water your plants properly. I admit to not being an expert at this (ADHD makes tasks like this interesting), but it seems better to err on underwatering, which is easily corrected, rather than overwatering, which is not.

5. **Use wine corks with a toothpick as plant labels** — This **gimmick** was described as a great way to recycle, but I don't know that our landfills are overflowing with wine corks. Corks are just oak bark, and will naturally, if slowly, break down in the soil or in a compost pile.¹⁹ Just use some popsicle sticks and let the kids use the corks in their craft projects.
6. **Use toilet roll cores as seedling pots** — Most people just throw out the core from a roll of toilet paper, but this is a true **hack**, as it uses up something that you are just going to end up throwing in either the trash or the recycling anyway. And it will break down in your soil and add some organic matter, as well.
7. **Use seeds from store-bought vegetables (e.g., tomatoes and peppers) to start your own garden plants** — This is just **bad advice**. Most supermarket vegetables are hybrids anyway, and won't come true from seed. Also, starting plants from seed is really hard work! If you're going to go through all that work, you might as well fork out for some high quality seeds.
8. **Use ordinary table salt as fertilizer** — This is just **really bad advice** because excess levels of salt can damage or even kill most plants. Maybe they were thinking of *Epsom* salt, which can be used as a fertilizer when properly diluted, because it contains high quantities of magnesium. (Nope, they actually list that one further down the list.)
9. **Make homemade weed killer with vinegar, table salt, and dish soap** — This one would actually probably work, because again, salt is really bad for plants, and vinegar will kill the leaves. But it won't kill the roots, which is I suppose why they are including salt. Also, in light of the previous "hack", is salt going to kill your plants or fertilize them? Leave the salt out and I'd be willing to call this a technique, but I'm not sure why weeds are even a problem, what with that thin layer of newspaper you put down all over the place. Again, this is just **bad advice**.
10. **Make fertilizer tea from your weeds to feed your plants** — This **gimmick** is just silly: take the weeds that you've just pulled up, put them in a bucket and cover them with water, wait for a few hours, get rid of the weeds and then water your garden with this miraculous, nutrient rich water. For one thing, you're just not going to get that many nutrients out of freshly picked weeds in a few hours. This is more like putting some leafy greens in a water

¹⁹The assumption that there is an entire privileged class that has so many wine corks that they don't know what to do with them says a lot about the person spouting this "advice" and their intended audience.

bath to perk them up. Second, you still have to get rid of the weeds. It would make more sense to just put the weeds in a compost pile, which is probably where this idea came from because compost tea is a real thing.²⁰

11. **Make holes to plant your seeds in by putting corks on the end of a garden fork** — This is not a hack, it's not a gimmick, nor is it even just a ridiculously terrible idea, it's also **physically impossible**. Most garden forks have tines that almost as wide as a wine cork is, so there's no way you're going to be able to stick a cork on there.²¹ And even if you could, this would just become bad advice because 1) not all seeds should be planted the same width apart, and 2) how difficult is it to make a hole in your garden soil to drop a seed in there? If your soil is that hard, you've got bigger problems that all the wine corks in the world aren't going to solve. For what it's worth, here's the entire process, in all its ridiculous glory:

*Sowing your seeds just got simpler! Rather than digging individual holes all along your garden bed, enlist the help of recycled materials to turn a garden rake into a makeshift sower. Just press an old wine cork onto each prong so that it's just just (sic) as long as you'd want your holes deep, then push the tool into the dirt. When you pull it back up, you'll be left with a row of holes ready for seeds.*²²

And that's it. There are plenty more examples, but I am out of space. This has actually been kind of fun, but I don't know if I have it in me to do another one of these, because there just aren't that many things that I know enough about to decide if something is an actual hack or not.

Sadly, a lot of this stuff comes from "lifestyle" sites where if something doesn't work, it really doesn't matter. You might be out a few bucks and a few hours of your time, but in the end does it really matter?²³ Probably not. It is notable, however, that if you google "brain surgery hacks" you won't get anything that involves wine corks or toilet rolls.

²⁰Although if you're going to go through all the trouble of making compost, you may as well just apply it to the soil and let your garden make its own compost tea every time it rains or you water it. Why are you going through extra steps?

²¹It's notable that even though this "hack" was accompanied by a picture of a garden fork stuck in the ground, there was not a wine cork to be seen anywhere.

²²Yes, this comes from Bob Vila's website. I know that there used to be a Cult of Bob Vila who thought he could do nothing wrong, but I beg to differ. The byline on his website is "Tried, True, Trustworthy Home Advice", but this bit of advice has not been tried and is definitely not true, which makes me question just how trustworthy the rest of the advice on his website is.

²³In that way, these lifestyle sites are a lot like religion: you sell a big promise, but when it fails to come about, you conveniently get to blame the user for doing it wrong.

What I've learned from this:

1. Ordinary people don't really know what is meant by the terms "hack" or "hacker".
2. Marketing people (who also don't understand what these terms mean) use them to sell stuff to people who want to feel like they are riding a trend.
3. Most things on the internet that are described as "hacks" are really just examples of really poor "journalism".

I put *journalism* in quotation marks, because while a lot of people might describe any sort of writing on the web as journalism, it really isn't. It's not like you need a degree and a license to call yourself a "journalist", alas.²⁴ While one of the strengths of the web is that anyone can publish whatever they like to it, in the absence of actual, careful research²⁵ on the part of the writer and actual, careful fact-checking on the part of a disinterested third-party, it's little more than junk at best, and harmful, dangerous junk at the the worst.

(I could—and probably should at some point—say a lot more about the importance of an independent free press to the functioning of a working democracy, but again, a different topic for a different zine. Suffice it to say that a *real* news source, be it newspaper, magazine, or television channel, is not afraid to print retractions when they get things wrong. If your news source is never wrong, it's not news, it's opinion, and you are being duped.)

²⁴It is, unfortunately, one of those words whose meanings have become watered down over the years.

²⁵By which I do **not** mean "find a bunch of YouTube videos and Facebook posts that confirm your own biases." Quite the opposite, in fact.

Chapter 3

Music (♪) in L^AT_EX

3.1 Standard Notation

I have a couple of guitars in the corner of my living room that I keep intending to dust off and play again someday. Of course, with the way my mind works, I thought I should also brush up on some music theory. Because I am an inveterate note-taker, but also have less-than-ideal penmanship, I thought I should figure out if it's possible to write music (and also guitar tablature) in LaTeX, and if so, how much work it would be.

As it turns out, there are a number of packages that enable you to include music in a LaTeX document.²⁶

3.1.1 wasysym

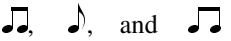








First, there is the `wasysym` (or Waldi Symbol font) package, which is basically a symbol font for LaTeX. It includes a lot of different symbols, including some interesting circles (○, ◌, ◌) that I could have used on a different project had I known about this then. It's a symbol font, so when it comes to music, what it offers is rudimentary: you can add an eighth note (♪), a quarter note (♩), a half note (♪), a whole note (♫), or two joined eighth notes (♯). In math mode, you can also add a natural symbol (♮), a flat symbol (♭), or a sharp symbol (♯).

I have to admit, while I like this, I'm not entirely sure what it's for. It's possible that it started out as one thing and ended up as another. That is certainly true of most of the projects that I've created. I'll investigate more later.

²⁶For much of this early research, I am highly indebted to Martin Thoma, who has an excellent introduction at <https://martin-thoma.com/how-to-write-music-with-latex/>.

3.1.2 harmony

There is also the harmony package, which offers up some additional symbols:

 (which is actually two symbols stuck together:  and ) , γ , and a whole lot of very tiny notes: , , and  and some very tiny rests: , γ , , and 

And it also has what I believe is chord notation (although I could be—and probably am—wrong; it has been a *very* long time), some of which can be quite complicated:

$$D \begin{matrix} 8-7-6-5 \\ 6-5-4-3 \\ 6-5-8 \end{matrix} 7$$

I didn't create that. (I don't even know what it means.) I just copied it verbatim from the harmony guide. In reality, it looks like this:

Harmony Example

```

1 %
2 \def\h#1h{\hspace*{#1em}}
3 \newcommand{\Str}[2][0.5]{\raise#1ex\hbox to
  → #2em{\hrulefill}}
4 \newcommand{\ST}{\h0.03h\Str[0.65]{0.27}\h0.03h}
5 \h0,5h\HH.D..8\Str[0,65]{1,2}7\ST6\ST5.8\ST6\ST4
6 \Str[0,65]{2}3.6\ST5\ST8
7 \Str[0,65]{3}7.
```

What this tells me is that the harmony package is very good at positioning things around other things. I may just have to tuck that in the back of my mind for later. The guide document for harmony is only five or six pages, but there is a lot of information to absorb there.

3.1.3 musixtex

And then there is the musixtex package. It makes use of a music environment, with your relevant code (of which there is a lot). I've copied this bit from the musixtex documentation:

Musixtex Example

```

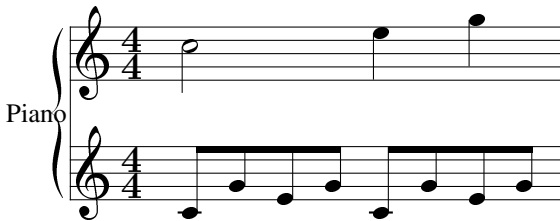
1 \begin{music}
2     \instrumentnumber{1} % a single instrument
3     \setname1{Piano} % whose name is Piano
```

```

4      \setstaves1{2} % with two staves
5      \generalmeter{\meterfrac44} % 4/4 meter chosen
6      \startextract % starting real score
7          \Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
8          \Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\ql
           ↪ l\sk\ql n\en
9      \zendextract % terminate excerpt
10     \end{music}

```

which produces this bit of music:



What strikes me most is that half of that code is not in any way intuitive. The package is well-documented, but at 166 pages, it's going to take some work to become proficient. There's no reason you *couldn't* learn this, but it's not something you're going to do overnight. Still, I think this might be the package I am looking for.

3.1.4 ABC

The abc package allows you to use the ABC language to create snippets of music directly in your LaTeX document, using the abc environment. Unfortunately, while I can get this to work in standalone documents, I have not been able to figure out how to incorporate it into a regular LaTeX document like this one. I obviously have more work to do to figure this one out.

3.2 Tablature

Tablature is a way of indicating which strings and frets to play on stringed instruments like guitars, basses, and mandolins. (There are arguments about whether it's worth using tabs, as they do leave out some information, but I won't get into that here. Suffice it to say that everything has its divisive issues. Personally, I find that if you already know the tune, tabs are generally enough.)

3.2.1 guitartabs

The first package I found is `guitartabs`. Unfortunately, this introduces a completely separate document class (called, *natch*, “guitartabs”) which means I can’t really use it here. (In general, I dislike it when packages do this, because it really limits the usability of their package. In this case, it makes it impossible to incorporate guitar tabs into whatever existing document you would like to include them in.)

3.2.2 songs

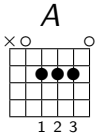
The `songs` package²⁷ says that it produces guitar *tablature*, but what it really produces is guitar *chords*. This should not be surprising, since it is really designed to produce church hymnals. I have very little use (none, actually) for church hymnals, but if you do, I recommend you look into this package because it is fairly powerful.

This code:

Songs Example #1

```
\gtab{A}{X02220:001230}
```

produces this chord diagram:

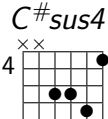


Like it? Here’s another:

Songs Example #2

```
\gtab{C#sus4}{4:XX3341}
```

produces this:



²⁷Which is available on SourceForge—see my later comments about that site.

Unfortunately, the `notes` package is incompatible with the `musixtex` package which I used earlier, as they both contain the commands `meterfont` and `transpose` and I can't figure out a way to make them play nicely together. My solution was to create a new document with the `standalone` class, create the chords in that document, and add them here using the `\includegraphics[]{}` command. (See the source code for how I accomplished this.)

This does point out one weakness of \LaTeX : it is possible to create a package with the same macro names as some other package, which essentially renders them incompatible.²⁸ It would be great if package developers would use some sort of personal prefix for their macros, but we are so far along in this game that I think it's far too late.

3.2.3 `musixtex`

As it turns out, this package can also be used to produce `tab`. However, because it was designed for classical music, you'll have to do some work to figure out how to use it for regular guitar tablature.

3.3 Lilypond

There is also a free and open-source software package (originally based on \TeX ?) which is excellent at setting music—Lilypond. I downloaded this several years and one or two machines ago, and remember thinking that it was big, powerful, and somewhat complicated to learn. “Well,” I thought, “we'll just pack that one along for later when we have time.”

It is now *later* and, well... here we are. I still haven't figured out Lilypond. But it is an exceptional program, and if you are interested in typesetting music, I encourage you to check it out.

3.4 Summary

As it turns out, incorporating musical notation into a text document (which was my original goal) is not that easy in \LaTeX . Most of the packages out there are either too simple to produce something useful like a music tutorial or even meaningful music notes. The `musixtex` package seems to have the most potential for something like this, but it is far from intuitive. (If `\Notes\ibu0f0\qb0{cge}\tbu0\qb0g\lq1` makes sense to you, I think we are definitely buying donuts in different donut shops.)

²⁸For what it's worth, this is possible with just about *any* bit of software.

Chapter 4

An Introduction to L^AT_EX

I realized that even though I've mentioned that the reason I created this zine was to learn how to use LaTeX, and even though I've mentioned the things I've learned about LaTeX while writing it, I've never really provided a basic guide for others who might be interested in learning L^AT_EX. So here goes...

A Caveat First, I am far from an expert in these matters. What follows is pretty much a listing of what I've gleaned from hours spent searching the internet and trying things out myself.

Second, some things will look differently and behave differently for you depending on variables such as the document class you are using (see below) and which other packages you have loaded. My rules for learning things like this are always:

1. Don't compare yourself to others. Your mileage can—and will—vary, because people learn things at different rates and in different orders.
2. Practice doesn't make perfect, but it does make things less shitty.
3. A willingness to experiment is your best guide.
4. You will get errors. Try to avoid getting the same error over and over again. Strive for better errors that you can learn something new from.

Also, if you have access to the source code so you can see how other people have done things, so much the better. (You can view the source code for this zine at <https://git.kjodle.net/kjodle/the-codex>.) In particular with L^AT_EX, it can help if you create an MWE (minimal working example) when working with new things, to rule out interference from other bits of code.

4.1 Files

\LaTeX uses plain files with a file extension of `.tex`. That's it! They contain plain text only and no binary codes or hidden formatting extensions. (Try opening any word processing document in a text editor and see what I mean here.) You can use any text editor, although a GUI is available for most operating systems.

4.2 Paragraphs and White Space

To start a new paragraph, simply skip a line. \LaTeX collapses white space, so if you are importing text from a text document, any lines that are adjacent to each other will be in the same paragraph. Additionally, multiple spaces will appear as a single space. For example, this code:

```

_____ White Space Example _____
1 This is the first paragraph.
2 This text, although it is on a separate line, is also part
  ↳ of the first paragraph.
3
4 We have skipped a line, so this starts a second paragraph.
5 This line is also in the second paragraph.
6
7 Readers will           not           see
8           all of           these
9 spaces.
```

renders like this:

```

_____ White Space Example _____

This is the first paragraph. This text, although it is on a separate line, is also
part of the first paragraph.

We have skipped a line, so this starts a second paragraph. This line is also in
the second paragraph.

Readers will not see all of these spaces.
```

4.3 File Structure

Every \LaTeX file has two parts:

1. A **preamble** in which you declare the document class and add any packages you may need, as well as set other variables such as the title and author.
2. A **document** environment which contains the actual text of the document.

If you are familiar with html, these correspond roughly to the `<head>` and `<body>` elements.

4.3.1 The Preamble

Within the preamble, you can declare the document's *class*, which is a description of the type of document you are creating. The most common classes are `article`, `report`, `book`, `memoir`, and `beamer` (for presentations). A typical class declaration looks like:

```
\documentclass[twoside]{report}
```

The class is described between curly brackets, but you can also include several options in square brackets. The above document class is a two-sided report. "Two-sided" means that it may have different margins, headers, and footers on right and left handed pages. Other options, such as paper size, are also available.

4.3.2 The document Environment

Your actual text goes in the document environment, which looks like this:

```
\begin{document}
  ...your stuff...
\end{document}
```

4.3.3 Publication Structure

Each \LaTeX document can be divided into a hierarchical structure consisting of the following sections:

```
Part
  Chapter
    Section
      Subsection
        Subsubsection
          Paragraph
            Subparagraph
```

To add one, use something like this:

```
\chapter[Books]{Books I Have Read}
```

Notice that we have the option [Books] which describes how this chapter will appear in the table of contents. To prevent a section from being numbered or appearing in the table of contents, replace this entire option with an asterisk:

```
\chapter*{Books I Have Read}
```

For best results, stick to the hierarchical structure shown above, as this is also how each section will be numbered. See the table of contents of this zine as an example.²⁹

4.3.4 Environments

This is where L^AT_EX shows its power, as environments are used to take care of typesetting tasks. Every environment begins with `\begin{<environment>}` and ends with `\end{<environment>}`. In fact, we've already seen one environment: the document environment, which encompasses our entire published document. Here are a few other useful ones:

Enumerate

`enumerate` is used to create numbered lists. They can be nested to create an outline. To prevent L^AT_EX from adding a lot of space between the item numbers, add the `enumitem` package and pass the [noitemsep] to the environment.

For example, this code:

Enumerate Example

```

1 \paragraph{The First Punic War}
2 \begin{enumerate}[noitemsep]
3 \item Carthage and Rome
4     \begin{enumerate}
5     \item Beginning of Foreign Conquests
6     \item The Origin of Carthage
7     \end{enumerate}
8 \item Operations in the First Punic War
```

²⁹I am a stickler about hierarchical structures because they represent logical, organized thinking about a subject. Not all subjects lend themselves to a perfectly hierarchical information structure, but we should always strive to be less disorganized (i.e., less shitty).

```

9           \begin{enumerate}
10          \item Outbreak of the War in Sicily
11          \item Capture of Messina and Agrigentum
12          \end{enumerate}
13 \item Events Following the War
14 \end{enumerate}

```

produces this output:

Enumerate Example

The First Punic War

1. Carthage and Rome
 - (a) Beginning of Foreign Conquests
 - (b) The Origin of Carthage
 2. Operations in the First Punic War
 - (a) Outbreak of the War in Sicily
 - (b) Capture of Messina and Agrigentum
 3. Events Following the War
-

Itemize

Similar to the `enumerate` environment, the `itemize` environment creates bulleted lists, which can also be nested.

As an example, we'll use the above example, but as a bulleted list:

Itemize Example

```

1 \paragraph{The First Punic War}
2 \begin{itemize}[noitemsep]
3 \item Carthage and Rome
4     \begin{itemize}
5     \item Beginning of Foreign Conquests
6     \item The Origin of Carthage
7     \end{itemize}
8 \item Operations in the First Punic War
9     \begin{itemize}
10    \item Outbreak of the War in Sicily
11    \item Capture of Messina and Agrigentum
12    \end{itemize}

```

```

13 \item Events Following the War
14 \end{itemize}

```

produces this output:

Itemize Example

The First Punic War

- Carthage and Rome
 - Beginning of Foreign Conquests
 - The Origin of Carthage
 - Operations in the First Punic War
 - Outbreak of the War in Sicily
 - Capture of Messana and Agrigentum
 - Events Following the War
-

You can also replace the bullets with any math symbol available in \LaTeX like this:

Bullets Example

```

1 \begin{itemize}[noitemsep]
2   \item[ $\Box$ ] First item
3   \item[ $\aleph$ ] Second item
4   \item[ $\triangle$ ] Third item
5 \end{itemize}

```

which produces this output:

- \Box First item
- \aleph Second item
- \triangle Third item

4.3.5 Math in \LaTeX

\LaTeX has a couple of different environments that are useful for typesetting math (`align` and `array`, but they get a little beyond what I want to cover here³⁰). In addition, there are other packages (in particular `amsmath` and `mathtools`) that

³⁰Maybe in a later issue? I can, if there is interest.

greatly extend the power of LaTeX to handle mathematical typesetting, but again, they are beyond the scope of this zine.

There are two types of *entry modes* for math in LaTeX. The first is **in-line mode**, which begins and ends with a dollar sign, and renders the math in the same line of text as the rest of the paragraph.

In-Line Math Example

The Pythagorean Theorem is $x^2 + y^2 = z^2$.

which renders as

The Pythagorean Theorem is $x^2 + y^2 = z^2$.

This is particularly useful if you want to include Greek characters in your text, because the code for the letter is simply the letter itself. For example, `&\alpha` renders as α . Need capital letters?³¹ Just capitalize it: `Gamma` \rightarrow Γ .

Math in LaTeX can also be shown in **display mode**, which renders the mathematics on a separate line. This entry mode begins with `\[` and ends with `\]`. If we change our example up above to this:

Display Mode Math Example

The Pythagorean Theorem is $x^2 + y^2 = z^2$

we get this:

The Pythagorean Theorem is

$$x^2 + y^2 = z^2$$

Installing LaTeX I realize that I haven't talked about how to install LaTeX, but that really depends on what system you are running. Trying to include every possible operating system could easily turn this from a zine into a book.³² The best approach is to search for your operating system + "install latex".

³¹At least for the characters where the Greek and Latin alphabets *don't* share a common capital character.

³²And a quickly outdated and inaccurate book, at that.

Chapter 5

What's to Like About Linux

5.1 Open Source

This doesn't come up for me every day as I use Linux, but one of my favorite things about Linux is that it's open source. That is, the source code is easily and freely available, provided you know where to look for it. This means two primary things:

1. I can download the software for free, and install and use it on as many machines as I choose to.
2. I can download the source code and make whatever changes I like. I can then keep them to myself, or release those changes to the world for other people to use.

We'll look at each of those two things in turn. But first, read the back cover and then come back here.

Let's start with point #1.

5.1.1 Open Source Software is Available for Free

The two main commercial alternatives to Linux are macOS (Apple) and Windows (Microsoft).

macOS is less widely available than Windows. The only way to get it is to buy an Apple product. I admit, Apple products are *gorgeous* and the hardware is usually designed to last for a long time.³³ But it is also *expensive*. I've done a bit

³³iPhones may be the exception to this.

of cost comparison and the cheapest Macintosh laptop I can buy is easily five times the price of the cheapest Windows laptop I can buy.

That actually makes Windows sound cheap in comparison. But is it?

Oh look, it's our only diversion.

We are in Late Stage Capitalism. That is, we have reached a stage of capitalism which is unsustainable on this planet alone. The entire point of capitalism is “growth” but when you are limited to an existence on a single planet that has no trade with other planets, that growth is limited by the amount of whatever resources that planet has. And when those resources run out—or it becomes impossible to make a profit by exploiting them—capitalism will collapse, and it will very like take billions of us with it.

These resources have always been *physical* resources—food, wood, coal, oil, rare-earth elements, and the labor of the working classes to extract and process them.

Because those resources are finite, we've seen capitalism go off in other directions to find other resources to exploit. This explains why rich white guys with entirely too much time on their hands want to ride their penis-rockets into space—because asteroids are out there and they're just full of iron, cobalt, and other metals just waiting to be mined and turned into cell phones.

But we've also seen capitalism go in a hitherto new direction: the digital direction. Capitalism is no longer in-

terested in just mining iron or coal; it's also interested in mining *data*. Data, it seems, is the one resource we can never run out of.³⁴

The primary kind of data they want is information about what we are interested in and what we want to buy, because if they know what we are interested in, they can then sell that information to advertising platforms who will slug your user experience full of advertisements for those exact things.

Order a ham sandwich for lunch online? That is now data. We know you like both ham and sandwiches. (Okay, this is really an *assumption*, but algorithms are built by people, and they are only as good as the people who build them. In short, algorithms make the same assumptions as the people who build them.)

Order a ham sandwich on Monday, but order a turkey sandwich on Friday? That is also data. Now I know lots of things about you:

1. You like sandwiches.
2. You sometimes order out for lunch.
3. You prefer pork-based products early in the week.

³⁴When I worked in retail, I was always told to push the extended warranties whenever I could. This was partly because the profit margin on them was incredibly high (extended warranties are basically a legal scam because there are so many terms and conditions you'll never be able to cash in) and “they're the one inventory item we never run out of.”

4. You prefer poultry-based products later in the week.

That is now four data points that can be sold to companies that will direct advertising at you for those items. And they will combine that data with other things they know about you (where you live, how old you are, your marriage status, your income, etc.) to come up with a list of stuff they think you'll *probably* be interested in and slug those ads in front of you.³⁵

Of course, they aren't selling that data to a local mom-and-shop sandwich shop down the road that cures their own meats and bakes their own bread according to a 100-year-old family recipe. They're selling that data to multi-million dollar corporations whose success will be built on the corpses of little mom-and-

pop shops.³⁶

As a result, when you think "I could really go for a sandwich right about now" you automatically think of the nearest Evil International Sandwich Shop and not the local mom-and-pop sandwich shop, simply because you saw an ad for the EISS, and when it comes to both hunger and money, your big huge mammalian brain depends on that little tiny reptilian bit at its core. You *think* you are choosing, but you are not. Like a rat in a Skinner test, you are merely being rewarded for pushing the button that capitalism wants you to push.

And this is what Windows does. It gathers your data and "shares" it with other companies it has paid agreements with. The price is low, but the cost is high.

Windows may appear to be low-cost or even free when you buy that new computer, but it does have a very high cost—your privacy. And because advertisers use some pretty sophisticated techniques to get you to buy their stuff, there is an additional cost—your ability to truly choose for yourself.

Which brings us to point #2.

5.1.2 You Can Alter the Source Code

This is really the greatest thing about open source projects: if you find something you don't like, you can change it. For example, if I want to change something about Linux, I can head over to <https://github.com/torvalds/linux> and become a contributor, or I can *fork* the software and create my own version.

That's a huge advantage, because it is not something that you can do with either Windows or macOS. But it comes with a huge cost: you have to know how to write code.

That phrase—"write code"—covers a *lot* of territory, because there are a lot of different programming and scripting languages and just because you know one

³⁵This isn't *exactly* how it works, but you get the idea.

³⁶Oh wait, that's already happening.

does not know that you know the rest, although people will often think that you do.³⁷ Knowing one does not mean that you know all of them, just the same way that learning Spanish does not mean that you now automatically know Urdu or Mandarin or even Italian. It takes time to learn the basics, it takes more time to become proficient, and it takes even more time to understand the code structure of the project you are looking at.

So yes, there is a cost, but if you decide to pay it, it has the additional benefit of making you more knowledgeable than you were before. Learning to code anything, even something as simple and straightforward as basic `html`, requires that you think logically and systematically about what you want to achieve and that you develop a systematic way of solving problems. Because, yes, nothing ever just works the first time you use it (if I had a dollar for every error message I've ever received while using LaTeX, I could probably afford to take a year off from work) and a *systematic* method for diagnosing and solving problems is a lot more efficient than just guessing.

A side note:

An optional benefit if you learn to write some code is that you can—and should—learn a revision control system, which will keep track of your changes and let you go back to an earlier version if you *really* screw something up, which you are bound to do at some point because really screwing something up is part of the learning process. I like and use `git`, which is included in the Ubuntu default repositories, but others such as `mercurial` and `bazaar` are also available. I like `git` because it's what I've worked with the longest and know fairly well, but that doesn't make it the best one; it's just the best one for me. They all have their advantages and disadvantages, and the point is not to find the perfect one,³⁸ but to find one that will meet your needs. Once you do, you'll wonder how you ever got along without it.

³⁷A long time ago I was a contributor to a theme for WordPress and worked as a (volunteer) moderator on its forum. Somebody wrote a post to complain that something he had attempted was not working the way he wanted it to and could we please fix it because, as he said "I'm sure it's just a coding thing", to which my (internal) reply was "Dude, if it's just a 'coding thing' then *you* figure it out". His implication was that we basically have a magic button that does things instantly, and we just sit around the rest of the time drinking coffee. The coffee thing (or any caffeinated beverage, really) *is* true, but the sitting around thing and the magic button thing are definitely *not* true.

³⁸The perfect one doesn't exist. The entire idea of perfection is one which has entered our culture through the door of religion, and it is an idea which we much abandon to regain our cultural sanity.

5.2 Containerization

Thanks to the Unix Principle, a Linux-based operating system is not one big piece of software, but rather, it consists of a number of smaller software packages that work together (more or less). That means that if you don't like the way something behaves and you can't get it to do what you want it to do through its preferences, you can often replace it.

For example, I became unhappy with Nautilus, the default file manager in Ubuntu. (The file manager is what you see when you open a folder on the desktop.) There wasn't anything particularly wrong with it—it just lacked a lot of features that I wished it had. Sure, I could learn to code my own file manager, or I could attempt to contribute³⁹ to the Ubuntu source code and get Nautilus to behave closely to what I was envisioning, but I didn't need to because there was already a file manager available that did what I wanted—Nemo.

Nemo is actually a fork of Nautilus (version 3.4, I believe) and includes a lot of features that were removed from Nautilus. It was developed as part of the Cinnamon desktop for Linux Mint, which was itself a fork from Ubuntu. Did I mention that Ubuntu is a fork from Debian? Yeah, open source family trees get pretty complicated pretty fast.

It turns out that installing Nemo and making it the default file manager is fairly simple. These are the commands you need:

```
1 sudo apt install nemo -y
2 xdg-mime default nemo.desktop inode/directory
   ↪ application/x-gnome-saved-search
3 gsettings set org.gnome.desktop.background
   ↪ show-desktop-icons false
4 gsettings set org.nemo.desktop show-desktop-icons true
```

What each line of this code does:

1. Installs Nemo (the `-y` flag automatically answers “yes” to any questions apt sends your way)
2. Makes Nemo the default file manager in Ubuntu
3. Disables the handling of the desktop by Nautilus

³⁹Sometimes open source projects become so large that they are difficult to contribute to—the equivalent of banks being “too big to fail”. Unless you are one of TPTB in the project, chances are you won't be able to make the kinds of changes you want to. But I have enough experience with this (I'm looking at you, WordPress) for a longish article or a shortish book, and I don't want to spend too much time here discussing it. Just know that it's a thing that happens.

4. Enables the handling of the desktop by Nemo

Removal is the opposite of installation:

```
1  xdg-mime default nautilus.desktop inode/directory
   ↪ application/x-gnome-saved-search
2  gsettings set org.gnome.desktop.background
   ↪ show-desktop-icons true
3  sudo apt purge nemo nemo*
4  sudo apt autoremove
```

1. Makes Nautilus the default again
2. Enables Nautilus to draw desktop icons
3. Uninstalls Nemo
4. Removes any dependencies used only by Nemo

As it turns out, this method allows an extremely high degree of customization while knowing very minimal code. It is possible, of course, to type those instructions wrong and completely screw something up, so it's good to double-check what you enter into the terminal. The terminal is extremely literal—it doesn't make guesses about what it thinks you meant to do (which is one of the reasons I find working with Microsoft Office such a miserable experience most of the time), it just does whatever you tell it.

An old joke that describes how the terminal thinks

A woman sends her computer programmer husband to the store to buy a gallon of milk. On his way out the door, she adds "If they have eggs, pick up a dozen". The store had eggs, so naturally, he came home with a dozen gallons of milk.

The joke, of course, arises from the fact that she never changed the object of the verb "pick up". What she should have said was "If they have eggs, pick up a dozen eggs, also".

Yes, this is how computers "think"—i.e., process inputs. **You have to be specific.**

Chapter 6

Ubuntu 22.04 (and Ubuntu 24.04)

I just got this bucket back together. I'm not going to let something tear it apart.

—Han Solo

Back in issue #2 I wrote about what a complete and utter disaster Ubuntu 22.04 had been for me and for many other users. I'm pleased to say that most of those issues have now been taken care of through numerous updates. But it has definitely been an uphill climb that left me questioning my sanity at times.

Most notable is that I still have not been able to configure Python2 on Ubuntu 22.04. Which is *not* a big deal really, as we all should have moved on to Python3 by now. (This is something that the powers that be at Python have acknowledged by deprecating Python2.) What I really needed Python2 for was PDF Booklet, which is what I use to turn all of these letter-half sized pdf documents into booklets that I can print and staple together.⁴⁰ I actually met the author of this package on SourceForge where it is hosted and he told me how to remove the Python2 dependencies, which I did. I figured out how to configure it as an installable (i.e., a

⁴⁰What PDF Booklet does—and does well—is handle the page imposition. That is, it puts the pages in the order that you would get if you took the staples out of this zine and examined the pages—you'll note that the first sheet has pages 40 and 1 on one side and pages 2 and 39 on the other side. This pattern continues until you get to the last page which has pages 22 and 19 on one side and pages 20 and 21 on the other. Page imposition is not at all complicated, but it never fails to amaze me how many people simply can't wrap their minds around it until they see it in action. I was one of them, once.

deb) package, but then I lost track of him on SourceForge and have no idea how to contribute to that project.

... *slides out soapbox*...

I'm going to take this moment while I have your attention to say that I detest SourceForge will all my heart. It's ugly, it places ads first and foremost in your user experience, it's ugly, it has a confusing interface, it's ugly, it's slow and clunky, and on top of it all, it's also ugly. It's like something designed by an alien who can only see half the visible spectrum and all of the ultraviolet spectrum. And it's been like that *forever*. Why is this thing still a thing?

... *slides soapbox back under the sofa*... *takes meds*...

Anyway, because I couldn't figure out how to get this on SourceForge, I just dropped it into my own personal git server at <https://git.kjodle.net/kjodle/pdf-booklet>. Enjoy.

As I write this, the release of Ubuntu 24.04 is less than a year away. Unlike the release of Ubuntu 22.04, I am not in the least bit excited. And I will **not** be upgrading immediately, if at all. (Ubuntu 22.04 has ten years of security releases,⁴¹ so I will be good for a while.)

I do have a spare machine that I can experiment with Ubuntu 24.04 on. The main problem is that I don't have a lot of time.⁴² So yeah, I can install Ubuntu 24.04 and *ooh* and *aah* over whatever is new and shiny (and I admit, "new and shiny" was a big part of the reason I was so eager to update to 22.04—I'm a human and our eyes are quite naturally drawn to "new and shiny"). But I don't have the needed time to fully experiment with it and determine if it is suitable to my needs (which range from "has to be perfect" to "can be just the right amount of shitty") or if it's another disaster-in-waiting.

And yeah, the emphasis in the promotional material will focus on the "new" and the "shiny" because, like I said, human beings are just naturally attracted to those things. But *new* and *shiny* don't always translate to *useful* and *functional*. That was certainly true of 22.04 out of the box.

All of which is to say that it will be a *long* time before I upgrade to Ubuntu 24.04. Canonical (the corporate interest behind Ubuntu) lost my trust with 22.04, and it's going to take a long time before they regain it. As the song says "I won't get fooled again". And that is assuming that they *can* even regain it. ("Meet the new boss" the other song goes; "same as the old boss".)

So yeah, a bird in the hand is worth two in the bush. Sometimes it's worth 22.04 of them.

⁴¹See <https://ubuntu.com/about/release-cycle> for more information.

⁴²If I did, you'd be reading issue #12 of this zine, rather than issue #3.

Chapter 7

Coda

7.1 What I Learned About L^AT_EX While Creating This Issue

7.1.1 Image Sizes

A few people have noted on Reddit that the images I included in prior images are far larger than I need them to be. This is true. I had a lot of things like this in issue #2:

```
\includegraphics[scale=0.5]{paper_cutter}
```

And it seems that the worst offender was this line:

```
\includegraphics[scale=0.13]{c128}
```

This is a problem because that `paper_cutter.jpg` image is 136kb in size and that `c128.jpg` image is 2.3 mb in size. Because these get included in L^AT_EX, which then handles the scaling, the resulting pdf file is rather bloated, which is the exact opposite of what you want when you are distributing something via the internet.

Because I am scaling the paper cutter image by 0.5, the resulting file size should be about a fourth of that, or 34 kb. And because I'm scaling the `c128` image by 0.13, the corresponding file size should be about 0.0169 of that, or roughly 40 kb in size. That's a huge difference.⁴³

⁴³Roughly, the size of your file should be approximately reduced by the inverse square of your scaling factor. But these are jpg files, which are lossy, so it's never exactly that amount.

The problem is that I always envisioned this zine as being a physical object, not a digital one. I only uploaded it to my git repository because this is a learning project for me, and I wanted to keep track of any changes I made.

But yeah, you should definitely resize your images before including them in any document you intend to distribute digitally. (Thank you, GIMP.) You'll notice that the image of the VT100 terminal on page 5 clocks in at a very sensible 79 kb. That's more like it.

7.1.2 Installing L^AT_EX Packages on Ubuntu

Method #1

Every once in a while I run into a package that I want to use with LaTeX that is not installed on my system. Alas, there is no easy way to do this, at least that I found. *Later*, I always thought. I'll figure that out later.

Later became *today* when I wanted to use the harmony package to produce some music symbols but couldn't, because it wasn't installed on my system. It took some doing, but I finally figured it out. Here's what I did:

First, I downloaded the harmony package from <https://www.ctan.org/pkg/harmony> and unpacked the archive. That was easy enough.

Next, I had to figure out where to put it. I ran this command:

```
$ kpsewhich -var-value TEXMFLOCAL
```

which gave me this location:

```
/usr/local/share/texmf
```

Following the directions I found at <https://jvgomez.github.io/pages/manually-install-latex-packages-in-ubuntu.html> I entered that directory using `sudo` and created the following directory structure:

```
/usr/local/share/texmf/tex/latex/harmony
```

I then copied my files over from the unpacked harmony files:

```
$ sudo cp * /usr/local/share/texmf/tex/latex/harmony
```

I then had to update the search path to make my system aware of the new package:

```
$ sudo mktexlsr
```

This should have been the end of it, but the `harmony` package requires a certain set of fonts to do its work. After a bit of searching, I found them at <https://ctan.org/tex-archive/fonts/musixtex-fonts>.

Fortunately, that package had a pdf document that described how to install the fonts. I started by copying the “fonts” folder over to the `texmf` directory:

```
$ cp fonts /usr/local/share/texmf
```

And then I updated the search path and then the font-map files with these commands:

```
$ sudo mktexlsr
$ sudo updmap-user --enable MixedMap musix.map
```

Like I’ve said elsewhere, I’m running a recently updated Ubuntu 22.04 system, and this worked for me. It may work for you, it may not. I can’t even guarantee that it will work for me next time. (I suspect having a generic folder name like “fonts” may cause me issues down the road if I ever want to install some other LaTeX fonts.) But if I got it to work once, I’m pretty sure I can get it to work again. We shall see.

Method #2

A perhaps easier (and certainly more *portable*) way of installing LaTeX packages is to add them to a local directory along the lines of:

```
/home/user/texmf
```

This is how I installed the `musixtex` package, and it is far simpler than what I have described in method #1.

First, I downloaded the package and unpacked it. Then I copied its contents into my `texmf` directory, and finally updated the filename database:

```
$ cp * ~/texmf
$ texhash ~/texmf
```

Mischief managed!

7.1.3 Miscellaneous Things

1. Need a little bit more control over things in a `verbatim` environment? Just add the `fancyvrb` package.

2. Need even more control than the `fancyvrb` package gives you? Try the `fvextra` package. (I used it because it very nicely breaks lines inside this environment.)
3. Notice that your footnotes are floating above the footer on some pages? Try adding `\usepackage[bottom]{footmisc}` to your preamble.

7.2 What I learned About L^AT_EX While Creating Something Else

For reasons I don't understand I went down an internet rabbit hole reading about the book *Flatland*, by Edwin A. Abbott. This is a book I had purchased years ago in my youth (thank you, Dover Thrift Editions!) but had never gotten around to reading. I found a copy in L^AT_EX at <https://github.com/Ivesvdf/flaflatland>. It was old—twelve years old, in fact—and it was set up as a single-sided A4 document. If you've been following this journey this far, you know that I'm pretty fond of booklets, and that I'm in North America, so everything has to be lettersize paper.⁴⁴

I downloaded it, and decided to play around with it to see how much I could make it look like an actual book. My original purpose for starting this zine was to learn how to typeset things in L^AT_EX, but it can be limiting since I've already figured out this format. Since I learn best from projects, another project was in order. This one fell into my lap at the perfect time.

7.2.1 Removing Page Numbers from “Part” Pages

In L^AT_EX, chapters can be grouped into “parts” using

```
part[ ]{ }
```

wherever you want a new part. (See the section on publication structure on page 22 for more information.) Surprisingly, these pages still have page numbers, which I just find odd. You can remove them by adding the `nonumonpart` package.

⁴⁴As an American citizen, I am bound by the U.S. Constitution to both completely disavow the metric system and be utterly confused by it, and to decry it as terribly confusing despite the fact that it is based on dividing and multiplying by the number 10. This is part of our constitutional duty to vehemently oppose anything which makes sense and also makes life better, such as universal health care. I don't know, something about eagles and gravy and guns.

7.2.2 Adding Additional Text to “Part” Pages

It’s fairly easy to *remove* page numbers from the “Part” pages, but it not nearly as easy as it is to *add* text to them. And it should be! In actual books, these pages often contain some sort of epigraph.

As it turns out, you can make the text an optional argument to the `\part` command by adding this to the preamble:⁴⁵

```
\makeatletter
\let\old@endpart \@endpart
\renewcommand\@endpart[1][ ]{%
\begin{quote}#1\end{quote}%
\old@endpart}
\makeatother
```

`makeatletter` changes the `@` to the “letter” category code so that the current document has access to package internal macros. `makeatother` changes it back to a letter so you can use it in your document.⁴⁶

`\let\old@endpart \@endpart` says “take the old value for `endpart` (which is part of the `part` function) and give it this new value that I’m about to describe”.

The rest of it (that is, the `renewcommand` part) redefines the `endpart` to now include a `\quote` environment, which is quite appropriate for an epigraph.

7.2.3 Adjusting the Line Spacing

LaTeX was designed to write documents; as such, its ability to fine-tune certain document parameters, such as line-spacing, is fairly limited out of the box.⁴⁷ But if you need something, chances are that someone else has needed it before you and has created a package that will do just that. In this case, the package you need is the `setspace` package.

Add that to your preamble, and you can adjust the line spacing of your document by adding either `singlespacing`, `onehalfspacing`, or the `doublespacing` command to your preamble.

⁴⁵As described by David Carlisle at <https://tex.stackexchange.com/questions/336361/how-to-write-text-after-part>.

⁴⁶This gets into the internal workings of LaTeX and so is far beyond the scope of this zine. However, there is some good information at <https://tex.stackexchange.com/questions/8351/what-do-makeatletter-and-makeatother-do> and at <https://www.tug.org/pipermail/tugindia/2002-January/000178.html> if you are interested. A complete list of category codes can be found at <https://en.wikibooks.org/wiki/TeX/catcode>

⁴⁷As it should be! Remember, the Unix Principle is to do one thing and do it well, not to be a Swiss army knife.

That gives you the versatility that you had with a typewriter. To set more precise line spacing, you can use the `setstretch` command in your preamble:

```
\setstretch{1.1}
```

which sets a line spacing equivalent to 1.1 lines. If you want to change the line spacing for just a portion of your document, use the `spacing` environment:

```
\begin{spacing}{2.5}
```

Your widely spaced text goes here.

```
\end{spacing}
```

This is a small, simple package, but one that I am sure I'll have a lot of use for down the road.

7.2.4 Include Page Numbers on “Chapter” Pages

I set up my version of *Flatland* to have the page numbers in the header and to completely suppress the footers. But I wanted those headers to appear on the chapter pages (you'll notice in this zine that they don't appear), because this is how the book was originally typeset back in the nineteenth century. As it turns out, the `titlesec` package makes this very easy:

```
\usepackage[]{}{titlesec}
```

```
\assignpagestyle{\chapter}{fancy}
```

For what it's worth, if you want to *omit* the headers and footers on a given page, just add

```
\thispagestyle{empty}
```

somewhere *after* the start of the page.

7.2.5 A Few Little Things

- Want SMALL CAPS? Wrap them in `textsc{ }`.
- Need to control the gap between the header and the rest of the text? Pass the `headsep` argument to the `geometry` package and set it equal to the amount of space you need (i.e., `headsep=12pt`).
- If you want to add a degree symbol to inline text, the simplest way I've found (so far) is to just pop in and out of math mode with this: $\text{\textcircled{0}}$ which gives you this: °

I almost forgot to add: If you are interested in seeing this project, you can view it at and download it from <https://git.kjodle.net/kjodle/Flatland>.