the codex

Life with Linux — A Zine

Typeset in LaTeX

Issue #004

Kenneth John Odle

June 14, 2024

Impressum

FYI, this is made in LaTeX using the report document class. It then gets exported to a letterhalf (5.5 in x 8.5 in) pdf, which then gets made into a booklet using PDF Booklet, which you can find at

`https://pdfbooklet.sourceforge.io/wordpress/`

The image of Linus Torvalds on the front cover is courtesy JericoDelayah from the WikiMedia Commons and is at `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. A list of topics I may cover in the future can be found at `https://git.kjodle.net/kjodle/the-codex/wiki/List-of-Future-Topics`.

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.

If you want to donate financial support for the creation of this zine (and all the hours of research that go into it), you can do so at

`https://paypal.me/kjodle` (Thanks!)

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.

In This Issue…

Chapter 1

The Final Salad Days

1.1 College, 2008

Bush II decided to tank the economy for ordinary people so that rich people could get richer.[1] I decided to go back into teaching (which, thanks to current conservative political policies[2] there will always be a demand for), which meant I needed to go back to college to renew my teaching license. But this is a whole other story for which I have run out of space (not to mention it has very little to do with Linux or even computers), so it will have to go in a different zine if I ever decide to write it down.

What I can say is that the first time around, I wanted to get a biology major and an English minor because I wanted to teach biology and English, and I thought (naively) that this was how things worked—you pick a major and a minor and you get a job teaching that major and minor.[3] But my biology advisor, a man who was many decades if not centuries my senior, advised against that plan. He felt that it would make me unhireable because it would look like I could not make up my mind between biology and English.

I did not realize it at the time,[4] but he was revealing his prejudice as a Biology professor. He was wrong, ultimately (schools absolutely love it when you can teach more than one subject as it provides for a lot of

[1]This is the second of three ”once in a lifetime” recessions I have lived through. Yay, capitalism! The rich get richer and everyone else gets poorer.

[2]Along with the asshole behavior of parents who approve of those policies.

[3]Turns out things don't work this way. Who knew? (You know who knew? Paul McCartney. "I look around me and I see it isn't so." I guess I failed to look around me.)

[4]I may not have realized it until just *now*, when I wrote this.

flexibility in scheduling), but his argument scared me. I was going to go thousands of dollars in debt for this degree (I was not smart enough to get a full scholarship, so I had to make up the difference with grants and loans—lots and lots of loans), and if I couldn't get a job, I wouldn't be able to pay back those loans. I would be sentenced to a life of penury, which is the very thing a college degree was supposed to protect against. So I agreed with him and forgot about getting an English minor.

It's time for a diversion!

If you complain about being stuck in a low-paying job, people of a certain stripe tell you that you should go to college. If you do go to college and then complain about being crushed under a tremendous amount of student loan debt, those same people will then tell you that you should have gotten a job that doesn't require a college degree.

What people consistently fail to remember is that college costs have risen disproportionately compared to the rate of inflation. From 1980 to 2024, the average price to attend a four-year college full time went from just over $10,000 a year to almost $30,000 a year when adjusted for inflation[5]—an increase of 180%. Whereas the state of California used to provide *free* college tuition back in the day[6] it no longer does—because then-governor Ronald Reagan wanted to punish the University of California for tolerating student activism.[7, 8] (Oh no! We've taught people how to think and now they're doing it!)

A well-educated populace is a public good and there is no humane reason why any economically successful nation should burden its youth with high levels of student loan debt. There are plenty of inhumane reasons, however, the primary one being that conservative politicians do not want a well-educated populace, as it is easier to exploit them for commercial gains if they have no idea what is happening to them. It's much easier to pin their woes on the boogeyman of "communism" despite the fact that they have no idea what communism actually is. Unless you have capital, capitalism is not your friend.

His second argument was that as a prospective biology teacher, I was required to get a group science "minor", which is in quotation marks

[5]https://www.forbes.com/advisor/student-loans/college-tuition-inflation/

[6]https://www.insidehighered.com/blogs/digital-tweed/tuition-free-college-yesterday-and-tomorrow

[7]https://calmatters.org/explainers/cost-of-college-california/#d6b48652-908b-4639-be19-3f09ecab02f9

[8]For more detailed information on rising college costs, see https://educationdata.org/average-cost-of-college-by-year.

because it was actually 36 credit hours (the equivalent of a major) rather than the 20 credit hours typical of an actual minor. As a result, I would have little time or energy (or money!) for another minor.

There are very solid reasons I got of out teaching, but unemployment makes us do strange things—hence the decision to go back to teaching which is really the only thing I'm good at. To renew my license, I needed to get eight credit hours in "a teachable subject" and I decided to take a couple of English classes, as that would both meet the legal requirements and also give me a chance to read and write for credit.

As they say, things happened.

At first I signed up for two English classes. But then I thought, that's eight credit hours. If I take three more I could actually get the full minor. Why not? So a couple of English classes became an English minor, which eventually became an English major.

Becoming a biology teacher required that I take a *methods* class, which is a class about... well, basically it's a class about how to *be* a biology teacher. It teaches you how to plan labs and field trips, and how to do things in a safe way so that nobody gets hurt, and it covers the specifics of teaching biology that were not covered in your regular education classes.

Becoming an English teacher required that I take *three* methods classes, however: one about teaching literature, another about teaching grammar, and a third about teaching writing. I remember very little about the first two. To be honest, I remember more about the graduate class I took in fairy and folk tales, because those tales evolve like living beings—which in a way they are. This was where biology and literature overlapped for me in the Venn diagram of my life.

The one thing I remember very distinctly about the writing methods course was that our capstone project had to be *online*. Oh wow, I thought—I've been creating websites for a while now. But I was worried. I had been hearing about how young people were so good with technology, far better than any previous generation, in fact. I was sure whatever they came up with would just blow my feeble old school attempts out of the water.

I could not have been more wrong.

Our instructor asked how many of us had any experience creating websites. Only three of us raised our hands: a woman close to my age, a college-aged woman, and myself. Half of the rest of the class looked perplexed and the other half looked mildly panicked.

Our instructor sent the three of us into a small computer lab at the end of the classroom to start making plans while she explained to the rest of the class how to create a web site. Of course, if you leave three students alone in a room, they're not going to get any work done—they're going to sit and talk. It doesn't matter what their ages are—work is just not going to be a priority for them. We didn't work. Instead, we sat and talked about how we got here.

As it turns out, the older woman and I both had experience creating websites from scratch and the younger woman (the only one of the three of us who was still on Plan A) had once set up a forum for her gaming community using phpBB, which is not quite the same thing as setting up a website, but close enough. It's better than nothing. She also spent quite a bit of time talking about "leetspeak" which had somehow passed me by—perhaps because I have never been much of a gamer.[9]

Still, this left some twenty-odd young people on the other side of that door who were utterly clueless about how to create a website, and were more than a little nervous about the prospect.

I mention all of this because it was generally assumed that young people were *so* good with computers, that they were *so* comfortable with computers, and why wouldn't they be? After all, they grew up with them. In fact, the term that got bounced around a lot in the education world at the time was "digital native". It was just assumed that young people could do anything with computers because they had grown up with them.

I had grown up with computers, too, only I did it twenty-five years before these kids did and the computers I grew up with did not have a GUI or a mouse. They made you think a little bit more than modern computers.

But it is not the presence or absence of a mouse or a GUI that makes you good with a computer. A lot of kids my age also had Commodore 64s and all they did with them was shove a cartridge in the back and play games. This in no way prepared them to know how to set up a spreadsheet in VisiCalc to balance your checkbook.[10]

I blame the educational industrial complex here, which I believe is where the term "digital native" came from. (And again, this is a feature, not a bug, of capitalism. Your main goal under capitalism is to just extract as much money as you can from the people around you, and it doesn't really matter whether you are doing the right thing or not. Your only value is determined by how much wealth you bring to the shareholders.)

Growing up with a computer in your house doesn't mean that you'll

[9]Everything I read about this makes me feel that is either something that is pretty cool or the most annoying thing in the world. I can't quite make up my mind.

[10]Which was one of the very first things I ever tried to do with a spreadsheet. It is not nearly as easy as one might think.

be an expert at anything and everything digital any more than growing up with a car in the driveway means you'll know how to drive it, let alone change the oil or rebuild the carburetor. It's a completely false assumption. The looks on my fellow students' faces proved that.

Still, most if not all of these other students figured it out. They created their lesson plans, they got them online, they presented a highly abbreviated form of them to the class, they got their final grade, and they moved on.

I did get mine online by doing what I'd always done—crafting html and css by hand.[11] And for a while it lived in a subdomain on my website but then *mobile* became the thing and I was not good at adapting sites to be mobile-friendly at that point (it's a piece of cake now[12]) and I also changed webhosts because I was not happy with some of the business decisions my then current webhost made and then I changed webhosts *again* because the dream host I had found (WebFaction) had been purchased by GoDaddy (which is a terrible company) and I eventually found a job outside of teaching because you can only eat for so long on a substitute teacher's salary, and I've let it go by the wayside. I'm sure I still have the files on a back-up drive somewhere, so I could theoretically get it online again (and in a mobile-friendly form) if I wanted to and had a long Saturday afternoon to devote to it.

But I'm not going to be a teacher ever again, so there's no point in that. That would be looking backward, not looking forward. And I don't like to get mired in the past, despite the fact that I have spent so many of these columns talking about what a delight it was. The fact is that *parts* of it were delightful and many more parts were utterly terrifying.

And if that's the past, who knows what the future holds? Terror or delight? Either way, like my former classmates, I'm moving on.

[11]Shades of *The Pushcart War*! One my favorite books from my childhood that shows that resistance is most definitely not futile.

[12]Also, there some sites that just *shouldn't* be mobile-friendly. Do you really want air traffic controllers directing airplanes from their phones?

Chapter 2

How to Be Good With Computers

Note: This originally started out as part of the the "College, 2008" section, but soon took on a life of its own. It didn't belong there, but it belongs somewhere. So here it is.

WHAT does it mean to be "good" with something? Especially with computers? The experience of being segregated into a little room almost twenty years ago because of something I knew while those around me didn't has marked me in some way. It's taken me a long time to actually figure it out, and I think it is that there really is no such thing as someone who is "good at computers". This is like "being good at sports" or "liking Asian food". The concepts of "computers", "sports", and "Asian food" are really too large to be considered as a single entity. There is really only "being better at this one very specific thing than everyone else in the room".

In general, the person who is seen as being good at computers generally has three characteristics that make them that way.[13]

First, they are not afraid to experiment. They know how to *undo* things, or they know to experiment on a copy of the file. This is easier to do on a computer than in the analog world because the physical costs are so low—it's basically just your time. You can make an unlimited number of copies. If I'm trying to figure out how long to cut a board to build

[13]I would argue that these three characteristics are common to people who are good at anything, whether it's computers, "sports" or cooking. These are essential characteristics of experts in general.

something, I don't have an endless supply of those boards. Ideally, I would like to make a single cut and get it done in one go and not have to go back to the lumber yard because I cut the board an inch too short.

Second, they are good at recognizing patterns. This means that they don't need to reinvent the wheel each time; they can look at a new problem and see if it is similar to an old one and whether it's possible to adapt an older, tried and true solution to this new problem. "The thing that hath been, it is that which shall be; and that which is done is that which shall be done: and there is no new thing under the sun."[14] All our new problems are just iterations of our old problems.

Third, they are good at searching the web for a solution. This is easier now than it was twenty years ago because the web is simply so much bigger than it was then. There is an art and a science to this, so much so that we used to use the term "google-fu" or "search-fu"[15]. But all this really means is knowing how to construct a search query that does not return superfluous answers. In the old days it was largely knowing how Boolean operators worked, and then selecting appropriate arguments for those operators. These days, artificial intelligence is probably going to screw this completely up. (Or it already has—I recently saw a screen clip of an AI bot recommending that you add half a cup of glue to your gravy to thicken it up. I'm sure this would work, but it would not be the gravy you are looking for.)

I can give a couple of examples here. At my old job, we received planning schedules from our customers on a weekly basis. Our material planner would print these out and enter their data into our system which would then give us an idea of how much production we needed to run each week.

One of our customers upgraded their planning software and what had been a twelve-page easy-to-understand report suddenly became a report running to well over a hundred pages that was full of irrelevant information and multiple empty columns.

I noticed that this information could be downloaded as a `.csv` file, so I created a spreadsheet that would allow you to import the `.csv` file, and which would then use VBA to get rid of all the information we didn't need, sort all the data in a way that made sense for our purposes, and then export the entire thing as a pdf file.

Because it worked (and worked *well*) I was lauded as a guy who is "good with computers" (which largely overlooked all the expertise my coworkers

[14]Eccleiastes 1:9

[15]Like "kung-fu". Get it? *Get it?* Sometimes we are tiresome people.

made use of every day as they did their jobs—which were largely computer based). The truth is that I knew absolutely nothing about VBA going into this. I just used those three characteristics—a willingness to experiment, an ability to recognize patterns, and a determination to search the web until I found the solution[16] I was looking for—and created something which worked.

But does this make me an expert at Microsoft Excel or at VBA? Hardly. I mean, it did to all those people who saw this spreadsheet in action. But I didn't view myself as someone who was good at Excel or VBA. I just viewed myself as someone with a dogged determination to keep experimenting until I got it right. To me, the idea of someone who is "good with computers" is *very* relative and not at all absolute.

Proof of this came in the form of an engineer that we hired out of his early retirement. He said he was not good with computers, but that he was adequate to the task required. Despite being "merely adequate", he showed me a trick that I have been using ever since.

One of my many frustrations with Excel, and indeed pretty much any spreadsheet program, has always been that if you need to edit the data at the end of the cell[17] it's always a two-step process. Either you click on the cell to select it and then click in the formula bar to move the cursor to the end of that data, or you have to click on the cell to select it, and then click again on the right end of that cell to move the cursor to the end of that data. (If you click in the middle of the cell, then that's where the cursor ends up. Who is in charge of this stuff?)

Just writing all that gives me a headache. It's no wonder people so often give the side-eye not just to Excel but to most Microsoft products. (As they should.)

But this engineer showed me that if you have a cell selected all you need to do is press the F2 key and the cursor will automatically move to the end of the data in that cell, where you can then use your keyboard to do what you need to do. As I use cursor keys rather than a mouse to navigate around a spreadsheet 90% of the time this is a huge timesaver for me.

This was sheer genius to me and yet I had never heard of it! If I had known this ten years before, it would have saved me so many mouse clicks and so much frustration.

In this instance, this engineer who viewed himself as anything but "good

[16]Or enough different solutions that I could put bits and pieces of them together until I got to where I needed to be. Every bit of software out there bears more than a passing resemblance to Dr. Frankenstein's creation once you look beneath the hood.

[17]Which is where I am most likely to make a mistake.

with computers" was indeed very good with computers, simply because he knew something I didn't.

Here's another example. At my current job we have an Excel form that we use every single day. It basically does three things: 1) it collects any findings we discover in an experiment, 2) it adds those findings to a database so that we have metrics to measure analyst performance by, and 3) it generates an email to send to the analyst in question so they make corrections.

The VBA that runs this form is absolutely huge—it's at least two orders of magnitude greater in size than any amount of VBA I have ever authored.

To my way of thinking the person who is responsible for the maintenance of this form is a *true* VBA expert. My level of expertise comes nowhere near theirs. And yet they still insist that they are not an expert, that they are just really good at searching for solutions on the web, at recognizing patterns (especially in error messages), and at experimenting until they get it right. (Sound familiar?)

I am still able to use VBA to do little bits here and there for other people—creating forms that generate a pdf and then automatically attach it to an email, for instance. I've done this a couple of times and the people on the other end have been suitably impressed, so I've added this skill to my resume. Imposter syndrome be damned!

The number one characteristic I've discovered that is shared by all people who are generally viewed as "good with computers" is *confidence*. Not knowledge, not skill, not experience. *Confidence*. It's true that confidence comes from knowledge, skill, and experience, but I know a lot of people who have all three of those things[18] but who still lack confidence and thus do not see themselves, nor are seen by others, as being good with computers. I've had to coach a lot of people like this over the years and while it's easy to give someone knowledge, or experience, or skill, it's almost impossible to get them to put those three things together into confidence. It's really something they have to gain on their own, and they either do or they don't. I'm not sure why this is.

THE most important thing about having confidence is that it helps you to keep going. This is how you can tell the difference between arrogance and confidence—confident people tend to keep going and arrogant people tend to give up.

[18]How can you *not* have experience with computers in 2024, when even Amish people have cell phones? It's because you can either choose to recognize that they are a part of your experience and roll with it, or refuse to acknowledge them and fight a losing battle against them.

And that's what I ultimately like about Linux—it encourages you to keep going. You don't get to a point where things are hidden away behind a proprietary brick wall. Everything is open source, right down to the core, and you can dig as deeply as you like. The only thing that's really holding you back (besides your attitude) is your time and money. But for me to even talk about these things would mean that I would have to do a deep dive into the many faults of capitalism, and that's not what this zine is about, although I do touch on that tangentially (or not so tangentially) from time to time.

That is really my biggest disappointment when it comes to computers. They were supposed to be the great equalizer, because everybody would have access to the same information and the same tools. But it hasn't worked out like that at all. We commodified everything. I look around me now and I thoroughly understand the reaction that the Taylor character had at the end of the original *Planet of the Apes* movie: "You maniacs!" he yells when he sees the ruins of the Statue of Liberty on the other side of the Forbidden Zone. "You blew it up! Damn you all to hell!"[19,20] Pretty much every pop-up asking me to join a mailing list or asking me to like a Facebook page,[21] every advertisement, every clickbait article, and every social media algorithm makes me feel this way.

Yep, we blew it all up so we could create value for the shareholders. We're not quite at a ruined-Statue-of-Liberty-on-the-beach point yet, but we are, I fear, very close to the end. We could have done better as a species, but we didn't. As Stephen King says of his generation, "we had a chance to change the world but opted for the Home Shopping Network Instead".[22]

The optimistic side of me says that it's not too late, while the pessimistic side of me says that it is, that we are already past the tipping point, but we just don't realize it yet. I'm not sure.

In the meantime, I'll keep doing what I'm doing. I'll keep writing, I'll keep striving to do better. And I'll keep going. What else is there?

[19]This is an authentically great film by the way. With a script cowritten by Rod Serling and based on a novel by Pierre Boulle (who also wrote *Bridge on the River Kwai*) and directed by Franklin J. Schaffner (who also directed *Patton*) how could it not be? I highly recommend it.

[20]Or as Paul McCartney might say, "I look around me and I see it isn't so." No, it is not so. It is *so* not so.

[21]Which is especially bizarre given that I haven't even had a chance to read their webpage yet. Why do I want to like something or sign up for something if I haven't even seen it yet?

[22]In *On Writing*—which is a great book to read, whether you want to become a writer or not.

Chapter 3

More Fun with bash

As I get older, I find that I want to spend less time doing repetitive tasks that need to be done, and spend more time doing the stuff I want to do, like writing.

As it turns out, Linux can help with that goal. More time writing and drawing and making music and making photographs is a good thing, and something I'm grateful to Linux for. The trick is, you have to be comfortable with the command line.

I'm a huge believer in having a workflow so that you are doing things consistently, and so that you can make gradual improvements to that workflow so you can get more done with less. Having a workflow means that if you are doing something wrong, you are consistently doing it wrong the same way. In which case, you only need to figure out a single fix and apply it to each mistake. If you don't have a workflow, you can screw up in many different ways, and have to figure out a lot of different fixes. Making mistakes is a part of life; making consistent mistakes makes fixing them a less miserable task. Linux makes it easy for you to do all of that.

3.1 bash Aliases for `git`

Back in issue #2 I talked about using bash aliases to make your life easier. I've also started using them with `git` as well. Here's what they look like:

```
bash aliases for git
alias gits="git status"
alias gita="git add *"
alias gitx="git add *.tex"
```

The first one just prints out the status of the git project that I'm working on. The second one will automatically add all files (except for invisible files) to the commit. Because I use LaTeX a lot, I also have the third one, which will commit any new or changed files that end in a `.tex` extension.

I rarely have invisible files in my git repositories except for the .gitignore file, which I seldom change, so I don't need a bash alias for it. I find it easy enough to type `git add .gitignore` on the rare occasion that I need it. But if I did want to add that file on a regular basis, I could just change that line to:

```
alias gita="git add * .*"
```

or I could just add a separate command for it:

```
alias giti="git add .gitignore"
```

Of course, if I were changing my *.gitignore* file that often, I would start to quite rightly question some of the other choices I've been making with my life.

3.2 bash Commands for *git*

It would be nice if we could do the same sort of thing for `git commit`, but we can't, because we need to add a message to our commit. In other words, it requires an *argument*. So for that, we need to add a *function* to bash.

As it turns out, this is pretty simple. It looks like this:

```
gitm(){ git commit -m "$1"; }
```

First, we start with our basic function, which is written like any other function:

```
gitm()
```

Now we add whatever commands we want between curly brackets. In this case we're only going to add one, which is the `git commit -m "$1";` bit. The only thing unique here is that we have a variable (`$1`) that references our first and only argument, which is the commit message we are going to add.

Once we have added all the files we need to our commit, we can then create the commit with something like this:

```
gitm "Updated section on bash aliases"
```

which is a *bit* shorter than typing

```
git commit -m "Updated section on bash aliases"
```

Admittedly, this doesn't save us a ton of keystrokes every time we use it, but if we make git commits on a regular basis, it saves enough keystrokes to make it worthwhile. Also, it's a shorter construction, so there's that much less chance of making a typo.

3.3 More about bash commands

As it turns out, you can add more than a single command to a bash function. For example, you can use this

bash function with multiple commands

```
cdl() {
     cd "$1" && ls -ahl;
}
```

This will change to whichever directory we specify with the `$1` placeholder, and then present a directory listing which shows all files, with human-readable sizes, in a long format. That may not be highly useful, but it's enough to give you an idea of how powerful bash aliases and bash functions can be.

As another example, I like to write rough drafts in longhand, on notebook paper. I find that I am more creative that way. The problem is that I intensely dislike being surrounded by piles of paper. (ADHD means that if I can't see something, it no longer exists. So my brain will only see whatever is on top of the pile.) Whenever I finish up a rough draft, I scan it to a "Drafts" folder, where it goes into a subfolder labeled for whatever projects it belongs to.[23] So that I can see everything, I use the `tree` command to

[23]Yep, there is a subfolder labeled "the codex" with drafts for this zine.

create a file which lists every single scan in that "Drafts" folder.

So far, so good, but running the same `tree` command consistently is not something my brain is set up to do. So I added this function to my `.bashrc` file:

```
drafts(){ tree $HOME/Drafts/ -R --prune >
 ↪  $HOME/Drafts/list.txt; }
```

What that command does is to go to that "Drafts" folder, run the `tree` command with the `-R` (recursive) and `--prune` (to ignore empty directories) options and then send the standard output to a file called `list.txt`. I look at that `list.txt` file whenever I am searching for something to write up, and I can see in an instant which rough drafts I can work on. My ADHD brain is pretty happy with this arrangement, as nothing gets buried in a pile of files, and I don't have a ton of paper sitting around.

For what it's worth, I also have a backup script (as I mentioned in issue #2) just for this folder. I added that command to the top of that backup script, so that before anything gets backed up to my cloud, that `list.txt` file gets updated and uploaded as well.

3.4 Reloading the `.bashrc` File

For any of these things to work, you need to reload your `.bashrc` file after you edit it. You can log out of your user profile and then log in again, or you can just go to the command line and type

```
source ~/.bashrc
```

And of course, there is also a shorthand version:

```
. ~/.bashrc
```

Chapter 4

The Right Ways vs The Wrong Ways

(i.e., The Hierarchy of Errors)

A lot of us grew up hearing that "there's a right way to do things and a wrong way to do things." I don't disagree that there is always a *wrong* way to do things, but like house maintenance, working on computers quickly teaches you that there are a *lot* of different wrong ways to do things. And despite what some people think, there is often more than one right way to do things.

Experience has shown me that not all wrong ways are wrong in the same way or to the same degree, and that the same is true of right ways. There may be multiple right ways to get something done, but some require less work and some require more work. It is not a black-and-white issue.

In the past few years, I've started thinking of things less in terms of a particular "right way" opposed to a particular "wrong way", and started thinking in terms of a spectrum of choices, some of which are obviously wrong (but wrong to varying degrees) and some of which are right because they work (but again, right to varying degrees).

What I have tried to do here is to create a hierarchy of "rightness" and "wrongness" as a way to organize my thinking on this subject; I can then jump in and discuss why things fall where they do. No doubt other people might have more or fewer distinctions in their hierarchy, or might have things in a different order, or might have different reasons.

My purpose here is to see if I am actually making any progress, or if I am simply doing things randomly, as adult-onset attention deficit disorder is apparently a thing in my life. In other words, do my changes move me up in this hierarchy, or do they move me down?

And, as we shall see, sometimes it's beneficial to do something the wrong way. You sometimes learn more by doing things incorrectly than you do by doing them correctly.

THE HIERARCHY OF ERRORS

- **Genius**
  - It's a true hack.
- **Right**
  - It works, and is considered a best practice.
- **Right*ish***
  - It works, but you have no idea why.
  - It works, but it requires you to rework some other pieces.
  - It works, but it's a bit of a kludge.
- **Wrong*ish***
  - It works in this specific instance, but not in all instances.
  - It works, but it's far more work than it should be.
- **Wrong**
  - It works, but it breaks things in weird places.
  - It works, but it breaks almost everything else.
  - It works, but it still manages to break a few local things.
- **Very Wrong**
  - It doesn't work and it breaks things in weird places.
  - It doesn't work and it breaks almost everything.
  - It doesn't work and it still manages to break a few local things.

Let's start at the bottom, and work our way up from there.

4.1 Very Wrong Ways

Very wrong ways are very wrong because not only do they *not* work, they take other things down with them.

It doesn't work and it breaks things in weird places. You may wonder why this is worse than "It doesn't work and it breaks almost everything else" but for me the answer is simple: it can be terribly difficult to find those weird places. When I say "weird" I mean that they may be obscure places that nobody looks, they may be distant from the current situation and apparently unconnected,[24] or they may be things that you don't have to rely on very often, so you may not discover that they are broken until days, weeks, or even months later.

It doesn't work and it breaks almost everything else. This is bad, but it is not bad as the previous example, because it has two advantages. First, because almost everything is breaking, those breaks are pretty obvious. Second, because almost everything is breaking, this provides you an opportunity to look at the overall structure of your project and examine how all the different parts are connected. There may be connections that you weren't aware of. You may realize that some things are connected that shouldn't be or that some things aren't connected that should be. Sometimes it takes a real disaster to point out the strengths and weaknesses of your system.

It doesn't work and it still manages to break a few local things. No complaints here. You have to undo what you did and maybe fix a few things, but you probably don't have a whole lot more to think about here.

4.2 Wrong Ways

Wrong ways may work, but they break other things along the way. As we shall see, this is not always a bad thing.

It works, but it breaks things in weird places. Again, the main issue here is that those weird places may not be obvious at first. You might use this technique, and it looks like it's working fine, but suddenly there is a person in Germany whose toilet no longer flushes properly. Or it works fine for you now, but in ten months *your* toilet no longer flushes properly. And because these two things are so separated in place (in the former case) or time (in the latter case) it can be difficult to connect the two things, and we might end up spending a lot of time going down rabbit holes when the

[24]But nothing is *truly* disconnected from anything else.

real solution is right there in front of us the entire time. We waste time and effort.

It works, but it breaks almost everything else. This is almost exactly like "It doesn't work and it breaks almost everything else" except that your solution *does* work. You just need to look at your overall system and figure out why everything else is going into meltdown mode.

It works, but it still manages to break a few local things. Even though this is listed as a wrong way—you are still breaking things, after all—this is not always a bad outcome to experience. It's possible that those few things that are breaking are breaking because they are weak. If you strengthen those items and then apply this technique, it turns out that this isn't actually wrong after all, it only seemed wrong at the time. In the end, you have a much better project that is much less fragile overall.

4.3 Wrong*ish* Ways

It works in this specific instance, but not in all instances. It works, so why is this way still wrong? Because it's not *universal* for all similar situations. If it works in *this one particular instance* but not similar instances, and you don't know why, then there is something about this particular instance that you are not aware of. This is a good thing if you're willing to chase down that unknown thing; it's potentially disastrous if you are not.

It works, but it's far more work than it should be. This is often a case of not having the right tools, or having the right tools but not knowing how to use them. If you need to dig a ditch, a shovel will work, but a backhoe works much better. All that time you spent working with a shovel is time you could have spent doing something else.

4.4 Right*ish* Ways

It works, but you have no idea why. I was very tempted to put this in the wrong*ish* section, and in some cases it may certainly belong there. Quite frankly, you *should* know why a technique works. Not knowing why can be dangerous, because you can assume too much about this particular technique. That may cause you to be a bit overconfident with it, and use it in a situation that doesn't really warrant its use.

It works, but it requires you to rework some other parts of the project. I admit, I was at a loss as to where to put this one. I guess it depends on whether you are using a kludge or a best practice, so I'm going to assume you are using a best practice. In which case, this shows you places that you were possibly *not* using something which is a best practice, and now you need to make those things better.

It works, but it's a bit of a kludge. A kludge is not always a bad thing (sometimes you have to work with what you have) but they are at best, inelegant, and at worst weighty and ugly. But they work for now, they don't break things, and they will last until you learn or can afford a better way. (I created a bit of a kludge when I couldn't figure out how to indent a bibliography entry.[25] Does it work? Yes. Am I happy with it? Not entirely. I'm 75% sure there is a better way to do this, but I haven't found it yet. But it works for now, and I've marked it as a kludge, so I know this is something that I can come back to later. At least I made this less weighty and hid its heft and inelegance by turning it into a macro.)

4.5 Right Ways

It works, and is considered a best practice. A best practice is one that has generally been accepted as the best way to do things because it produces results that are better than the results achieved by other methods, with either minimal or zero negative side-effects. This is a good thing. A best practice is a best practice because it's proven itself. It's not perfect (hence it's a "best practice" not a "perfect practice"), but you can count on it to get the job done. And because it is a best practice, when things go pear-shaped, it's most likely because of something you've done, but if it isn't, there will probably be a lot of people who are *very* interested in helping you.

Unfortunately, sometimes a best practice is arrived at that for no other reason than "that's how we've always done it and nothing has exploded yet." That's not great, but still...have a fire extinguisher handy.

[25]You can see it in action in this commit for a different project: `https://git.kjodle.net/kjodle/Notes-on-Python/commit/d4f93ec00f1e1078b1cfcb3aacd3481eb82bb0cd`.

4.6 Genius Ways

It's a true hack. As I said way back in the first issue, I define a hack as "an appropriate application of ingenuity"[26]. True hacks are rare and often small, whereas false hacks are all too common, and the internet is littered with them[27]. A false hack only resembles a hack; like the wizard in *The Wizard of Oz* who says he's a wizard and looks like a wizard, but is not an actual wizard.

If you find a true hack, enjoy it, preserve it, and help to disseminate it.

[26]See `http://www.catb.org/ esr/jargon/html/meaning-of-hack.html` for more information.

[27]See issues #1 and 3 for lots of examples.

Chapter 5

Not Another PDF Scanner

Way back in issue #1 of this zine I talked about my workflow for scanning documents because I am trying to be as digital as possible.[28]

In that article "A Scanner Darkly, but with a workflow" I mentioned that I used one piece of commercial software (VueScan) because it did what no FOSS software that I knew of at the time could do: work with my all-in-one printer/scanner and also sort pages effectively when my scanner's ADF[29] does not duplex (i.e., it does not flip pages over to scan the other side). While it is great software, and I did not mind paying the $100 for a one-year subscription to it (the software company behind it is pretty much a father and son team), I didn't like being dependent on it.

The reasoning is simple. If a company decides to stop producing a product, that's it; you're done. I used to have a great plugin on my WordPress sites that added social media sharing icons to each post. The company that made it got bought out by Oracle. You might think this is a great thing, because Oracle is a big huge company with a lot of resources. But when big huge companies buy small independent companies, they are often only interested in one or two of their products, and let the rest go. And this is exactly what happened. Oracle suddenly decided they weren't going to support this plugin and it just stopped working. The company's webpage for the plugin redirected to an Oracle page that basically said "fuck off" and little more. No explanation, no recommendations of similar plugins, nothing.

[28]Issue #1 is only three issues ago, but considering that I published it in 2021, it *seems* like a long time ago. I really need to get my act together and get these out on a more regular basis.

[29]Automatic Document Feeder

At least when FOSS software projects get abandoned or the original developers get better paying jobs delivering pizza, there is always the chance that someone else will take over the project. Better yet, you—yes, *you*—can donate money to the project to help support it.

I first found out about NAPS2 (`https://www.naps2.com/`) because I had downloaded a book from the Internet Archive[30] and the pages were very, very yellowed. (It had been scanned from a pulp paperback printed on cheap paper with a high acid content. How seldom we plan for the future!)

I was looking for a way to lighten the background of the pages so that it would be easier to read. My usual solution for this would be to edit the pdf in GIMP, opening each page as a separate layer. I could then figure out the settings for one page, convert that into a script (GIMP is scriptable!), apply that script to every single layer, and export the entire thing as a pdf, remembering to tick the box that says to export layers as pages, and also to do it in reverse order.

That's not a huge amount of work, but still—it's work. Surely, there has to be a more automated way to do this, no?

I searched and I searched, and I was rewarded for that search. Someone mentioned that a program called NAPS2 had this very feature. The name didn't hurt at all—at this point in my life, I am very much in favor of naps, unlike the five year old version of me.

5.1 Adjusting the Image Quality of a Scanned Book

How do you clean up a scanned book from the Internet Archive? The way I would normally handle this would be the GIMP method I described earlier. But that's a lot of work for a book I just want to read and be done with. (No archivist work for me here.)

The workflow for this is fairly simple. First you import your pdf using the "Import" button, then you select all the pages and click the "Image" button. The options are pretty limited: you can adjust the brightness and contrast, adjust the hue and saturation, or you can sharpen. It also has an option called "Document Correction" which is great if you are scanning hand written notes and need to add a lot of contrast. (This doesn't work so greatly in the case of a badly yellowed book, unfortunately.) You just make

[30] *Inherit the Stars* by James P. Hogan, which you can read at `https://archive.org/details/inheritstars00jame`

your corrections, and also make sure to tick the box that says “Apply to all pages”—something I often forget to do.

You’re probably not going to get a perfect book back, because the options are pretty limited when you’re correcting an entire book at once. The trade-off is that you pick your settings once, and then NAPS2 handles all the work while you go get yourself a cup of coffee—or take a nap.

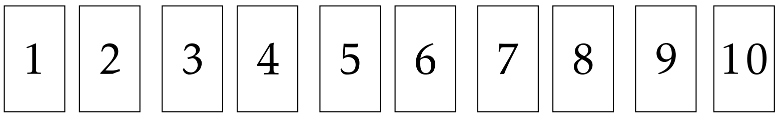
5.2 Interleaving

NAPS2 also has an “interleave” feature, which meant that I didn’t need to use `pdftk` to do that.[31] Scanning longer two-sided documents suddenly became a lot easier.

The only problem was that NAPS2 offered *four* versions of this command: interleave, deinterleave, alternate interleave, and alternate deinterleave. These are very neatly contained under the “Reorder” icon in the main menu. I knew one of those was what I needed; I just needed to figure out which.

I’m a scientist, so I experimented. I took five sheets of scrap paper, wrote the odd numbers 1-9 on the front side and the corresponding even numbers 2-10 on the back side. If you flipped through them, you would see something like what you see in figure 5.1.

Figure 5.1: The document as originally drawn

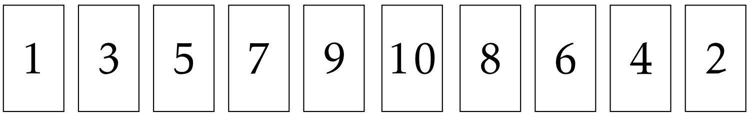


Using this scanner means every page has to be scanned twice—one time for the front side, and a second time for the reverse side. As a result “upside-down” has a couple of meanings here. My scanner’s ADF accepts pages with the side you want to scan face down. It then flips them over when it scans them, which rotates them around the y-axis. Because it flips them over along the y-axis, I only have to spin them around the z-axis to get them in the proper orientation to scan the other side. This should seem self-explanatory, but it often isn’t—I had to use the scanner at work the other day and it took me three attempts to get it right.

[31] To be fair, this is a feature which VueScan also eventually added at some point.

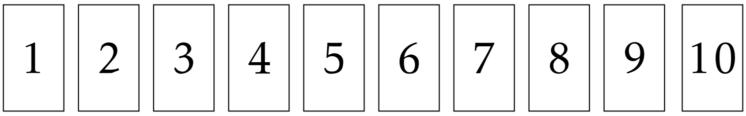
Because I wrote on both sides of each side of paper in order to emulate a double-sided original, I scanned the pages, and then spun them around the *z*-axis and scanned the other sides. And because I am scanning these face down, the even numbers end up in reverse order. So I ended up with a pdf that looked like figure 5.2.

Figure 5.2: The document as originally scanned



That's progress, but it's not the progress I wanted to make. I tried all the different options available under the "Reorder" icon, and finally figured out that "Alternate Interleave" would produce the final pdf that I want, which you can see in figure 5.3.

Figure 5.3: The document after applying "Alternate Interleave"



If that looks like the original document, it is definitely not an accident; it is by design. Print that out and you get something close to the original.[32]

5.3 Adjusting Images

Even though NAPS2 was designed to be a pdf scanner, it also has the ability to save individual scans as images. Even more importantly, because each scanned page is basically an image, you can also edit each page as an image by double clicking on it, where you get editing options like crop and rotate, in addition to the ones I mentioned earlier. This is pretty handy if you're scanning something like a manual that has different sized pages, or is printed on large sheets and folded into a box so that you have to scan it in

[32] I say "close" because a scan is never the equivalent of the original. It is a reflection, an imitation. But it is not the same. Every time we copy an analog object, we lose something. Replicative failure is a thing.

sections,[33] or a package that has care instructions on one or more sides.

If you're wondering why I keep banging on about manuals, it's because I do keep them. For years I kept them all in a large three ring binder filled with page protectors that I could slip them into. It was big, and it was awkward, and I didn't dare grab it the wrong way or I'd have manuals all over the floor.

At some point, I realized that most manuals are available in convenient pdf form from the manufacturer's website, so I started downloading those, making sure the pdf was identical (or identical *enough*) to the original, and then tossing the original in the recycling. But for those that aren't—yep, I scan them.

Summary: In short, NAPS2 is everything I need in a document scanner. It gives me some of the editing features of GIMP, has a simple interface (you can create different profile for each kind of scanning you do, which is *very* handy—I have about a dozen) and it just plain works. I recommend it.

[33]Every piece of furniture I've ever assembled has instructions like this, but I've run into quite a few manuals that are miniature versions of this, like the earbuds I wear on my daily walk.

Chapter 6

Chemistry in LaTeX

I used to be a science teacher back in the day and typesetting anything for chemistry was not all that easy. We generally just wrote and drew everything by hand. It would have been easier and much neater if I had known anything at all about LaTeX, but alas, I did not. I do now, though.

As it turns out, people have written a number of different packages over the years to help with this. Let's take a look at three of them.

6.1 Package `mhchem`

The `mhchem` package is useful for typesetting chemical equations and reactions and has a fairly intuitive interface, making use of a `ce` environment. For example, to typeset this equation:

$CO_2 + C \longrightarrow 2\,CO$

we would simply use this markup:

```
\ce{CO2 + C -> 2 CO}
```

As you can see, numbers placed after a letter are automatically formatted as a subscript. To format them as a superscript, we just use a caret (^) before the number. For example, `\ce{CrO4^2-}` produces $CrO_4{}^{2-}$. Pretty nifty, huh?

Note that the superscripts and subscripts are not stacked; this is the preferred method according to the IUPAC[34] Green Book. But if you want

[34]International Union of Pure and Applied Chemistry

them to be stacked, mhchem has this option:

```
\mhchemoptions{layout=stacked}
```

which will give us stacked superscripts:

CrO_4^{2-}

It also does a pretty nice job of rendering fractions. `\ce{1/2H2O}` gives us $\frac{1}{2}H_2O$. That's not the preferred way according to IUPAC, however. To accomplish that, you should use `\ce{(1/2)H2O}` which gives us $(1/2)H_2O$.

You can also do some fancier things. Here's another example from the manual:

```
\ce{Hg^2+ ->[I-] HgI2 ->[I-] [Hg^{II}I4]^2-}
```

which will give us:

$$Hg^{2+} \xrightarrow{I^-} HgI_2 \xrightarrow{I^-} [Hg^{II}I_4]^{2-}$$

If you need something fairly straightforward, the mhchem package is for you. It has a simple, intuitive interface and it does a great job. I wish I'd known about this when I was teaching.

6.2 Package chemformula

The chemformula package is similar to mhchem in many respects, but is stricter about how certain items are input. In return, it has more options to customize the output.

Like mhchem, it's pretty intuitive. To write the chemical formula for copper(II) sulfate pentahydrate, we would use code like this:

```
\ch{CuSO4 * 5 H2O}
```

which produces

$CuSO_4 \cdot 5\,H_2O$

The most notable difference between chemformula and mhchem is that chemformula can distinguish between different types of input, which are separated by a space. That means that in our example above, there are four parts: the copper sulfate part, the asterisk part (which `chemformula` renders as a dot), the "5" part, and the H2O part. `chemformula` can then detect whether each input is a formula, a stoichiometric factor, an arrow, etc., and format them accordingly.

You can also use math mode in `chemformula`. For example, this code:

```
\ch{$2n$ Na + $n$ Cl2 -> $2n$ NaCl}
```

gives us this reaction:

$$2n\,\mathrm{Na} + n\,\mathrm{Cl_2} \longrightarrow 2n\,\mathrm{NaCl}$$

We can also write the names of substances underneath them by using a ! and two pairs of parentheses, like this

```
ch{!(sodium)($2n$ Na) + !(chlorine)($n$ Cl2) ->
!(sodium\ chloride)($2n$ NaCl)}
```

which gives us this:

$$\underset{\text{sodium}}{2n\mathrm{Na}} + \underset{\text{chlorine}}{n\mathrm{Cl_2}} \longrightarrow \underset{\text{sodium chloride}}{2n\mathrm{NaCl}}$$

We had to use spaces inside the parentheses so that the package will know how to format these separate types of input. Also, because a space delineates different inputs, in order to get that space in "sodium chloride" we had to escape the space with a backward slash.

Again, there are lots of options to customize the output. Here's one with fractions:

```
\ch{3/2} (vertical fraction) \quad
\ch[frac-style=xfrac]{3/2} (diagonal fraction)
```

which gives us:

$\frac{3}{2}$ (vertical fraction) $\quad$ ³⁄₂ (diagonal fraction)

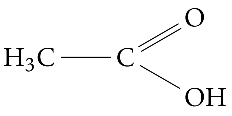
Like I said, this one operates a lot like `mhchem`. If `mhchem` works for you, there's no need to look further. But if you need more control over the appearance of your formulas and equations, `chemformula` will give you a lot of that control. It's also very well documented.

6.3 Package `chemfig`

If you need to draw chemical structures, then `chemfig` is the package for you. It uses a `chemfig` environment, and loads `tikz` if it hasn't already been loaded.[35]

6.3.1 Basic Syntax

The syntax is remarkably simple. You start with the first atom in a molecule and work outwards from there. For example, take a look at this at this drawing of acetic acid:



which is created using this code:

```
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```

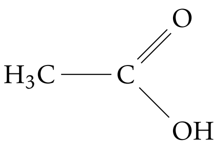
It's pretty easy to see what's happening inside the `chemfig` enivronment.

1. First we create the H_3C —— using `H_3C-`.
2. We then add the second carbon atom `C`.
3. From this atom, we branch by placing each branch inside parentheses.
4. We then angle those branches relative to the *x*-axis by placing the angle inside square brackets after a colon. In this case, we are specifying 30° angles.

We can also use predefined multiples of angles of 45° by omitting the colon, so that this code:

```
\chemfig{H_3C-C(=[1]O)(-[7]OH)}
```

produces this figure:



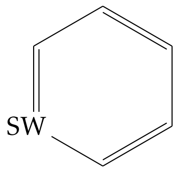
[35]See issue #2, § 7.2 for some basics about drawing with `tikz`.

Note that angles are always specified with regard to the origin and to the horizontal, regardless of where they start. In other words, [1] = $1 \times 45° = 45°$ and [7] = $7 \times 45° = 315°$.

6.3.2 Ring Structures

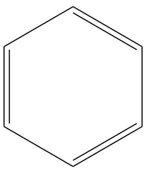
We can also create ring structures pretty easily by using an asterisk at the beginning of our definition. I've created some examples below, where the syntax should be fairly easy to understand. (And notice the use of `[,0.75]` to change the length of the line of the bond to the functional groups.)

Rings always begin with the atom in the southwest corner, which I've labeled here:



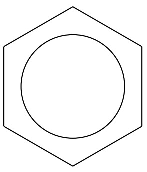
Benzene with double and single bonds:

```
\chemfig{*6(-=-=-=)}
```



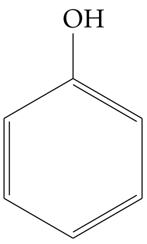
Benzene with a ring inside:

```
\chemfig{**6(------)}
```



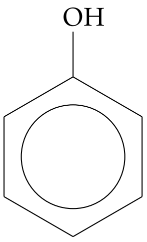
Phenol with double and single bonds:

```
\chemfig{*6(-=-=(-[,0.75]OH)-=)}
```



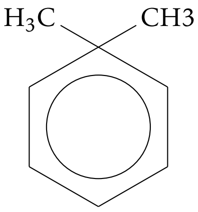
Phenol with a ring inside:

```
\chemfig{**6(----(-[,0.75]OH)--)}
```



Cumene with a ring inside:

```
\chemfig{**6(----(-[:150,0.75]H_3C)(-[:30,0.75]CH3)--)}
```

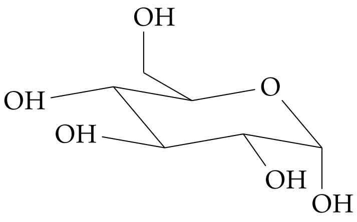


6.3.3 More complicated structures

Once you understand the basics, it's fairly straightforward to construct even more complicated structures. Here are some examples.

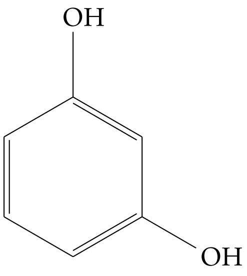
Glucose:

```
\chemfig{?(-[:190]OH)-[:-50](-[:170]OH)-[:10](-[:-55,0.7]OH)
-[:-10](-[6,0.7]OH)-[:130]O-[:190]?(-[:150,0.7]-[2,0.7]OH)}
```



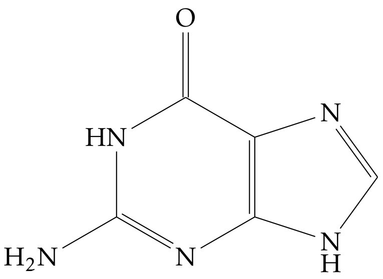
Xylene: (Or as IUPAC would say: 1,3-dimethylbenzene.)

```
\chemfig{*6(-=(-OH)-=(-OH)-=)}
```



Guanine:

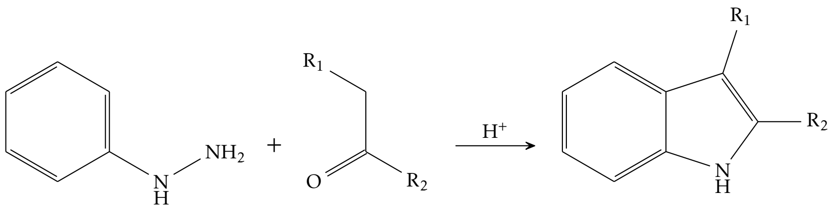
```
\chemfig{*6((-H_2N)=N-*5(-\chembelow{N}{H}-=N-)=-(=O)-HN
-[,,2])}
```



Note the use of chembelow to place the third hydrogen atom directly below the nitrogen atom, as we typically don't show the single bonds to hydrogen atoms in cases like this.

You can even use chemfigure to show reactions, using a scheme. Here's the Fischer indole synthesis that I took right from the gallery in the manual:

```
Fischer indole synthesis:
\setchemfig{atom style={scale=0.75}}
\schemestart
\chemfig{*6(=-*6(-\chembelow{N}{H}-NH_2)=-=-)}
\+
\chemfig{(=[:-150]O)(-[:-30]R_2)-[2]-[:150]R_1}
\arrow(.mid east--.mid west){->[\chemfig{H^+}]}
\chemfig{*6(-=*5(-\chembelow{N}{H}-(-R_2)=(-R_1)-)-=-=)}
\schemestop
\setchemfig{atom style={scale=1}}
```



I had to use setchemfig to scale the reaction to 75% so it would fit on the page, and then its use again to reset the scale back to 100%. (Although the latter is a moot point, as we are done now.)

This is an easy package to master, and really fun to use. I encourage you to try it.

Chapter 7

Coda

7.1 What I Learned About LaTeX While Creating This Issue

7.1.1 How to Get More Font Sizes

One of the things that has bugged me for some time about this zine is that I was not creating the cover in LaTeX. The reason for that was simple: I didn't know how when I started this project. But we're now at the fourth issue, and so it was time to finally figure it out.[36]

One of the issues that I encountered is that your standard document classes give you a limited number of font sizes, and I needed the title to be fairly large—larger than I could get with the `\begin{Huge}•\end{Huge}` command, anyway.

As it turns out, if you are using Type 1 fonts, you can just use the `fontsize` command to make this work. This is what I used for the front cover:

fontsize example

```
{\fontsize{50}{60}\selectfont \textbf{the codex}}
```

The first number (50) represents the font size, while the second one (60) specifies the line spacing, which is generally irrelevant if you are only

[36]If this cover looks a little different than previous covers, this is the reason why. But they will look like this going forward.

using this on a single line of text.[37]

7.1.2 Adding Space at the Top of a Page

Another issue that I ran into when creating the cover in LaTeX is that I needed space at the top of the page, before the text. Normally, I use something like `\vspace{50mm}` for something like that, but any `\vspace` is automatically deleted at the top of a page. The solution is to use it with an asterisk, like this: `\vspace*{50mm}`.

7.1.3 Drop Caps

Do you want drop caps in your LaTeX document? Then use the `lettrine` package. It's well documented and has lots of options. Just be careful not to overdo it. Drop caps can be overwhelming if used too often.

You can also be really fancy by using custom images. In this case, I used `y.eps` to create this drop cap. I probably didn't choose the best font for this kind of thing, but you get the idea. And you also can see what I mean when I say you shouldn't overdo them.

[37] I found this at `https://tex.stackexchange.com/a/716/245702`.

Chapter 8

Afterword

It's been a while, hasn't it?

I'm gratified that this zine has a small but dedicated fan base who have a lot of great things to say about it. The feedback that I have gotten has been nothing short of amazing. I appreciate every single bit of it.

I apologize that I can't get these out on a more regular basis. When I was younger, I wrote all the time, but as I get older, I find that I write more slowly and often with great difficulty. Part of that is no doubt my current job (which I was going to write about that in this issue, but I ran out of space, so it will have to wait until next time) and part of it is. . . well, all the things that go along with just trying to make it in this day and age.

And a large part of it is no doubt because I have short bursts when I get quite a bit done, and then long periods where I get nothing done. This is not new—one of my college professors said that I tend to "run hot and cold". Some people have told me that this sounds like bipolar disorder, and others have said that this is a natural part of the creative process for some people. I suppose there isn't any reason that it couldn't be both.

Another reason is that I find I write best when I can work longhand. There is just something about the feel of pen or pencil on paper that really gets my creative gears going. Once they get going, I can usually switch to working digitally without too many problems.

I have a website for all my zines, which you can visit at `https://just13.click/`. I used to have a mailing list, but then Mailchimp decided to get rid of their TinyLetter service because why provide a public good for people when you can make money instead? If you want an email notification of when I produce a new zine, just send me an email at `wolfgangswishlist@gmail.com` and let me know which zines you want to hear about.

I also have a list of topics I intend to cover in future issues. I've provided a link to it on page 2. If you have ideas for things you'd like me to talk about, you can send me an email at the above address.

Thanks,
—Ken