^Z
Ctrl-ZINE

The Future Is Ours

...should we choose to accept

The best...

> geek rag
> fan pub
> coffee table nix-zine
> Smol Web conversation starter

...since the 6502

Issue 5 - Vol 1

About ^Z

Ctrl-ZINE (^Z) is a Ctrl-c.club/Smol Web collaborative zine that celebrates tech and the Smol Web. Started in March 2023, it runs a monthly issue, where anyone can download a PDF version and a pre-folded PDF version for home printing. No digital format of the content is maintained on a Website whatsoever. Some of the topics within these issues range from Smol Web protocols and communities (ActivityPub, Tildeverse), Web-adjacent protocols (Gopher, Gemini), alternative forms of communication (HAM radio, IRC), snippets of code, artwork, and anything tech-related that is an expression of self.

Those who contribute to ^Z are passionate about what they share. They want what is best for Us, the citizens of the Web. With that, anyone with that same passion is welcome and encouraged to contribute to future issues. Further info can be found in the Editorial section of this issue. May the Smol Web live forever!

Editorial

Ctrl-ZINE

ZINEHEAD Press
e-mail: zinehead@fastmail.com

Contributors

~lettuce
~nttp
~giggles
~singletona
~loghead
~calamitous

CLI BYTE: "SSH (secure shell) - secure command line access to remote Linux systems"

Hello, and welcome, to the highly informative, endlessly useful, recklessly optimistic, and unintentionally insightful Issue 5 of Ctrl-ZINE. In the crowd are new and old members of Ctrl-c.club, a few blips and blobs from folks not in the pubnix world, but all with the shared vigor and tenacity of those participating in a Web “born once more” - pulling away from centralized outlets more and more every day, maintaining their presence and persona online with a much smaller footprint. Decompressing from brands and influence, in exchange for friends and collaboration. Where money isn’t the decision-maker, but YOU are.

These pages include handy bash scripts, reflective and (non-)linear trends and movements with one’s personal Web presence, recalling the utilitarian nature of RSS, a handful of words on AI’s potential. . .potential. and much more!

We hope you find something you like. And that your days stay filled with joy and fast WiFi!

Your compiler and geek,
~loghead

Bash alternatives to cloud products and platforms *by ~lettuce*

2023-06-14

Most web products from file hosting to chat to banking software are made up of a tech ‘stack’ within a large software ecosystem. These are built for ‘scale’ and designed to be administered by a company providing you a more or less turnkey product at various levels of services. You pay for increasing levels of convenience, either with your private information being tracked, or with recurring expenses. This could mean paying annual fees for a ‘creative editing suite’, downloading and running a health tracking app that invisibly sells your data to data brokers, or accessing a social media account that sells your data to advertisers. And there are many more examples you can imagine or read about in the news.

Lately a number of folks in my online community have been discussing the problems of the platform Discord, from issues of centralization, monetization of community conversation and labor, accessibility issues, to the inability to archive or save discussions or content. Some folks have gone back to using a combo of blogs and the old chat system IRC. In my own communities I’ve cut back in my participation in Discord and created alternative self-hosted forums or piggy-backed on Mastodon instead.

Inspired by a CLI asciicinema recording by software and hardware developer Phil Hagelberg on how to use IRC I started to think more seriously about the idea that many cloud services could be replaced by some Linux software or some lines of Bash code gluing programs together.

First, I want to address some straightforward question or critique someone might have. Namely, why use the (Linux) command line instead of a simple web platform? Another criticism of this

approach might be that using the command line could appear to just be retro nostalgia, or unnecessarily complicated.

To answer these: I think using tools that don't cost much or any money, that we or others can modify and share, and that we can combine together to meet our needs is empowering. I have a bicycle where I can fix a flat tire, replace the chain, and do minor maintenance like adjust the breaks. Sometimes I spend money at the bike shop such as when I needed generator lights installed and wasn't confident in my own work. The bike is what I use to commute to my studio, and how I get around town. And yet I feel okay working on it, and the knowledge gained from trying lets me step in and fix something when I need to. Though I'm not afraid to get extra help at a bike shop or from knowledgeable friends. This isn't a perfect analogy, but it'll do.

Likewise, working with computer tools and software doesn't need to be intimidating. You can learn a bit at a time, try things out, find tutorials, and look for community to help you along the way. It's also a good way to resist commercialization. Rather than buying 'products' and the need for incessant upgrading, annual subscriptions or throwing out old products to get access to the latest products, you can opt out. Like riding a bike command line and text user interface software can be beautiful, elegant, luxurious or just minimally works. With some basic Bash knowledge you can get pretty far.

I'll admit that some Free, Libre and Open Source Software can be ugly or clunky. I am also sympathetic to resisting pure retro-nostalgia, but I don't think continuing to use the command line in 2023 (or whatever year you are reading this) is simple nostalgia. Bash and the shell predates GUI software, and while I won't make a definitive prediction, it could even potentially outlast it. The shell never disappeared, and the amount of command line software has increased exponentially in the past

number of years. And most of it continues chugging along to work year after year. It's not that uncommon to be reading a command line software man(ual) page and it lists the year in the 80s! And it's still useful.

On Linux, our basic automation tool is the command shell. As opposed to cloud software and GUI software it's often much easier to develop and certainly to glue together command line software. The term 'glue' here means to combine command line software together in various ways, sometimes envisioned by the developer but other times not. The software will have ways to take input, to be modified or configured, and has standard ways to produce output. All this allows it to be composed together with other command line software. We still don't have a great way to 'glue together' GUI software in this intuitive way that we weave together software in Bash. Using Bash to glue tools together is such a fundamental advantage on Linux systems because it was intentionally built in from the beginning to Unix.

Glue code is often considered to be a form of 'duct tape programming.' It's fast. And glue code solutions might be considered a 'hack' approach, not necessarily implying the original term hacker here. And while this could imply that glue code doesn't last long it could also be thought advantageously as well. For the same reason that cloud services and platforms are black boxes where you input money or privacy/data and get out a simplified output or software product, Bash and other glue codes let you pick and choose, customize, see its innards, test your own idea by typing it and running it in the command line REPL. You continue to refine it until you find a solution, and then you can automate your solution, with scripts, cron, and the like.

Without further adieu, I present some sketches of ideas, recipes and speculative ideas on how to glue together your own

alternatives to FAANG and other startup products in the command line. Some of these are easy peasy for those new to the CLI. Others will require a bit more knowledge and experimentation.

Some of these will run on your own computer. For others you'll want to access a server, either because you need to 'sync' with someone(s) else, to store info remotely that can be accessed by multiple people or other reasons. You could join a tilde community or you can…

Run your own server

For many of these solutions, if you have your own server, either a spare old laptop, raspberry pi, or a remote server, you won't be as reliant on cloud services.

For an easier-to-configure server for this kind of thing, try yunohost.

```
yunohost
```

Setting up a server is beyond the scope of this article, but there is lots of documentation online if you do a search, or read the yunohost website.

Word Processing

Nano or WordGrinder are simple and attractive text user interface-controlled word processors.

Alternatively, text editors such as Micro, Emacs, Vim, Neovim are old faithfuls.

Live collaborative text editing

Use ttyshare or tmate or even tmux with ssh to create a shared terminal session. Then both open your text editors.

Alternatively, use Vim’s server capability.

Version control and collaboration (not ‘live’)

Git or Subversion plus a shared remote server, perhaps one you set up with yunohost.

Git server

You can use Gitea.

Or even simpler, bare git on a server, see:

Idiomdrottning’s How to host git repos on their Gemlog.

=> gemini://idiomdrottning.org/hosting-git-repos How to host git repos

minimal calendar program

    when

Synced calendar

    Khal

Pick a date

Put a spreadsheet on a server that others have access to editing. Or use email. Or host a form on a server using cgi. Have people select the best option. Schedule an email to you or all at the end date with the results. You could set it up with cron or just mark on your calendar to check back on a certain day.

Todo list

I save my todo list as a textfile. There's also todo.txt project.

Radio

Pyradio is excellent.

Image editing

Imagemagick is incredible. Lots of recipes are available online.

Image browsing online and offline

I use chafa to browse images in the command line. If I'm going through directories of images I use fff, a vim-like file manager. Pressing i over an image file will open it inline overlaid in the terminal.

The terminal emulator Terminology can also show images inline in the terminal. For example tyls is like the linux ls command setup to display images of all files.

For browsing the web in the command line there are a number of great programs but w3m has the w3m-img plugin that renders image in the command line. Add the -H flag to get 'high quality' images.

links text browser has the -g flag that enables graphics mode in the command line.

Image sharing

I have a simple bash script that generates html image galleries that I host on my web server. I use imagemagick to resize.

cyclenerd has an example called gallery.sh that automates this.

Alternatively you could use nextcloud to share images. I haven't tried yet so you'll have to explore on your own.

Social media

If you use mastodon, try toot.

twtxt is a minimalist social media protocol like a minimal distributed twitter (sorry for the birdsite comparison!). You can use a web client or browse and post in the command line.

File sharing

Looking for a dropbox or wetransfer alternative? Try The Null Pointer.

Or alternatively, upload to your own server. For sharing files on your own server, use scp.

For backups, rsync.

Forums

On Ctrl-c club tilde we use the iris command line forum software with hundreds of users on a single shared server. We love it.

Weather

This is a fun category. You can check the weather a few ways.

```
curl wttr.in
```

There are some other options you can pass in too.
Or use ansiweather.

Conclusion

Many of these solutions, systems and recipes are barely more complex than using a cloud service alternative. And they won't be mining your data, selling your info to advertisers, or trying to sell you additional services. With a little bit of elbow grease they can be put to good use. Help is often a search engine query away, or why not try posting on IRC? Many of these programs can be tailored to your own use-case. These programs are almost always free and some take donations. Beyond initial setup (if any), these programs also eschew all of the advertisements, pop-ups alerts and notifications and other cruft that contribute to mental exhaustion while using some of the commercial platforms these programs are replacing.

This also doesn't need to be an all-or-nothing affair. Pick and choose what works for you. That's the beauty of having access to free and open source software.

I hope you find some solutions to your own needs, and glue together your own software ecosystem.

Thirty Year Circle *by Singletona082*

1993

I had recently transferred from public schooling to a school for the blind, and had barely learned of the internet from a combination of PBS, and seeing a thing on one of mom's soap operas where the text was eighty point font so the audience could read the one sentence, or word, or whatever that was supposed to make whoever flip out over whatever was going on.

2023

I've just witnessed Reddit follow Twitter into a black hole of capitalism devouring its user base for the sake of Profits. Between looking for alternative platforms such as kbin and Lemmy that seek to emulate the experience of Reddit I'm reminded of the Tildaverse.

At a glance, these two incidents have nothing really in common. Yet for me there is a disturbing feeling of deja-vu. Both are from points in my life where everything is both incredibly static, yet also completely in flux. On the one hand, Eleven-Year-Old Me was completely unmoored from a physical situation where my scholastic needs weren't
met. On the other hand, Forty-One Year Old Me is in a situation where my social and informational needs haven't been met.

I suppose one could call it less a perfect circle, and instead call it a resonate spiral where events don't line up, yet have a similar feel.

1993

Child-Me is having to unlearn terrible preconceptions on how computers are used, which amounts to 'watch a guy flail away incoherently at a keyboard and things happen.

2023

Adult-Me is having to unlearn a lot of dependence on corporate-backed entities that had taken a lot of the drudgery out of computing for the sake of the lowest common marketable unit. This has amounted to ‘I can't let Google remember my passwords, and I have to relearn how the
forum works while I flail around the same way I laughed at the non-savvy thirty years ago.'

Thirty years ago I sat in front of a monitor wondering what I would make of this, but knowing it was something important, and thirty years later I sit in front of a much more powerful box just as clumsy with a command line interface as I was then, and yet still out of my peer group the one most willing to try even if I end up breaking something in the process.

Thirty years, and I'm still that same kid pecking away at a keyboard wanting to use a toolkit I barely understand to try making a text game for people I'll never meet but have a strong connection with.

It is not a circle. Computers, batteries, and the speed of information are orders of magnitude more than it was. This means even going back to an environment that detractors could call

chasing nostalgia, or the millennial version of boomer cane-waving isn't the same. We each have far more space to work with, on a machine that can do more in less time, accessible from devices that fit in our pockets if we want.

Thirty years ago, I wanted a state-of-the-art 486 tower that probably is the size of a mini-fridge that was made by a mass market manufacturer tapping into a rising tide of demand. All for the sake of playing with the internet and games made by companies riding what was then considered either a fad or the cutting edge.

Thirty years later, I'm looking at a device made by one guy that is being made in limited runs, is low powered device that can easily fit in my pocket all for the sake of logging into a terminal connection on a volunteer-operated computer that's there purely for the love and appreciation of fellow enthusiasts.

I wonder where walking this circle will get me in thirty more years.

RSS (news) feeds and how to tame them *by ~nttp*

2023-06-14

As everyone watches yet another social media website implode (bet no-one will remember which one I mean in a couple of years), people are talking about RSS again, as a way to keep up with news without relying on a corporation. Trouble is, plenty of people (still) aren't very familiar with RSS, or at least don't know how to get going. And my bet is that at least some people want to understand, not just be pointed at yet another online app and told "use this".

For those who don't know: RSS is a kind of newsfeed (another popular kind is Atom). That's simply a way for websites to let each other know about updates such as blog posts or press headlines. At least that was the idea at first. Then some smart person realized it can be used to let people know about updates, too.

This raises a couple of questions: first, why do you need to automate that. If you only follow like five websites, you can put them on the browser's bookmark bar, and click each tab in turn while sipping coffee. But once you have fifty or more? It would be a lot easier if you could click a button and get all the latest updates on a single page, nicely sorted.

Second, you might be wondering why you need to do anything special. Can't your browser simply load each website in the background and see what's new? Well, no, because computers are dumb. They need the data in a rigid, formal structure they can parse easily, because what looks easy to humans is cryptic for computers, and the other way around.

So you have these newsfeeds, usually of the RSS or Atom variety. Used to be, browsers could detect their presence automatically and show a nice button you could click to subscribe, but they removed this ability. Nowadays most sites show a link for it; you've probably seen the orange icon with the radio wave symbol.

Follow this link. If you see a nice list of entries, you're all set; look for the button that tells your browser to start tracking it (some of them still can). But if it shows a bunch of source code instead, don't panic! There's an app for it.

Okay, there's a lot of them, but Liferea is not so different from Readrops for example. Add feeds to them using the link, update all of them at once, then read at leisure.

You can even go offline to read the saved entries. Try that with a web app!

There are other tricks. You can import and export feed lists using something called OPML. You can add the site's address directly and let the reader detect existing feeds, if any (like browsers used to). You can even tell the feed reader to check for updates three times a day or whatever, but that to me seems like overkill to avoid pressing a button. Worse, it can be easily abused to hammer websites that already have enough trouble handling traffic.

Speaking of which: nowadays many websites ask for your e-mail address instead, so they'll find out who you are and spam your inbox. (Ironically, Thunderbird can read RSS feeds as easily as with an e-mail newsletter.) Others have their newsfeeds well hidden, like YouTube, or remove them entirely, like Twitter. An app like Fraidycat can still ferret out the desired updates.

But if you like this RSS thing and want to see more of it out there, your best bet is to spread the word: let people know it's an option. Thank you.

AI-80: A Graphing Calculator for the World's Information *by ~loghead*

Not dissimilar to how Steve Jobs felt PC's were a "bicycle for the mind", I'd say that AI, OpenAI, ChatGPT, etc., are likely the tangible, nuts-and-bolts, point-and-prove incarnation of a graphing calculator for the world's information. Much as the TI-80 graphing calculator can do damn near any numerical formulation, AI can (or, if it cannot, it soon will) be able to take *most* of what is documented the world (or, on the Internet) and "do something" with bits and bobs of what is being asked of it.

It is also a stark reprieve from social media fanaticism, and the bubble bullshit that all of it was. With it's slow and anti-climactic decline, and more or less living up to what everyone knew it was all along - a bubble market that had zero sustainability nor longevity (think: Yahoo/AOL = Facebook/Twitter - which is a formulation/comparison made to exhaustion by so many in the past decade).

So as we exit an era where history rhymes (but doesn't repeat) once more, new and improved (and authentically original) technologies take it's place. And though I, personally, will not delve too deep into AI (much as I sometimes wish I had avoided the bandwagon of the WWW when hearing of it in the 1990's), I know that it (AI) will make a profound impact on all things in the world. Impactful in a way that is both good, bad, and everything in between. Hence is the price of progress, and the realities of a "thing" (or, a protocol/service/anything) existing in the world with which we live.

Hacking finger *by ~giggles*

Hey folks, some day in the past I was talking with guys on #ctrl-c and I dont even remember why but someone said something about the program finger, before this I never had heard about finger before, its seems cool thing to share information in a shared environment like ~tildes, this is a direct quote from wikipedia about finger:

The program would supply information such as whether a user is currently logged-on, e-mail address, full name etc. As well as standard user information, finger displays the contents of the .project and .plan files in the user's home directory. Often this file (maintained by the user) contains either useful information about the user's current activities, similar to micro-blogging, or alternatively all manner of humor.

In our case we will focus on the **all manner of humor**, at least a single manner of humor is enough :| the idea behind this text it to make the finger output dynamic. As we know by the previous description the finger will display contents in the **.plan** file in the user home, as a quick test I just created a file with "Hi Ho!" content and used finger on my self (lol):

```
giggles@ctrl-c:~$ echo "Hi Ho!" > .plan
giggles@ctrl-c:~$ finger giggles
Login: giggles                          Name:
Directory: /home/giggles                Shell: /bin/bash
On since Sat Jul  1 14:05 (CDT) on pts/5 from tmux(1308688).%8
    1 minute 12 seconds idle
On since Wed Jun 28 18:14 (CDT) on pts/9 from tmux(1308688).%5
    1 hour 49 minutes idle
```

```
On since Sun Jun 18 16:32 (CDT) on pts/15 from tmux(1308688).%0
   4 days 5 hours idle
On since Sat Jul  1 13:59 (CDT) on pts/19 from tmux(1308688).%7
   3 hours 52 minutes idle
On since Sat Jul  1 13:56 (CDT) on pts/86 from tmux(1308688).%6
   3 hours 43 minutes idle
On since Sat Jul  1 13:09 (CDT) on pts/98 from xxx.xxx.xxx.xxx
   1 second idle
On since Tue Jun 27 12:57 (CDT) on pts/99 from tmux(1308688).%2
   4 hours 16 minutes idle
No mail.
Plan:
Hi Ho!
```

Ok, it worked as expected, the .plan file is on our output, but I wonder if we can use a cool thing instead of just a static file, before trying anything I looked in documentation and found about the ~/.fingerrc script, and wow it does everything I wanted, if this exists then the output of the script will be used instead of default finger output, so I just tried this but it dont work ~.~

Ok, lets look on finger man page and see if there is something which help us:

```
giggles@ctrl-c:~$ man finger
...
HISTORY
     The finger command appeared in 3.0BSD.

Linux NetKit (0.17)              August 15, 1999              Linux NetKit
(0.17)
```

I just kept the useful part here, because the main reason the script didn't work is that finger on ctrl-c server is the BSD version which come packaged in Linux NetKit and I was looking on GNU version documentation duh!, But I have an idea to overcome this and reach goal. We just to use a fifo instead of a regular file and monitor it for readings. In theory the finger will open

the file, read and then output. So we just need to write to our fifo to give finger the dynamic content! To test this I just created my .plan file as a fifo and tried to write "ohaio" on the fifo before using finger.

```
giggles@ctrl-c:~$ mkfifo .plan
giggles@ctrl-c:~$ echo "ohaio" > .plan
```

Then from another terminal (tmux panel for real) just finger the user as following

```
giggles@ctrl-c:~$ finger giggles
Login: giggles                          Name:
Directory: /home/giggles                Shell: /bin/bash
On since Sat Jul  1 14:05 (CDT) on pts/5 from tmux(1308688).%8
...
No mail.
No Plan.
```

Hmm, our echo still blocked trying to write on fifo and the finger said there is **No Plan :(**, ok time to get more info, so this time we run finger again using strace (could be ltrace too)

```
giggles@ctrl-c:~$ strace finger giggles
....
lstat("/home/giggles/.plan", {st_mode=S_IFIFO|0664, st_size=0, ...}) = 0
write(1, "No Plan.\n", 9No Plan.
)               = 9
exit_group(0)                           = ?
+++ exited with 0 +++
```

Hmm, it's using lstat in the file and just going straight saying there is no plan, I guess it's checking if its a regular file. To make sure this is the case we can just download this version of finger, there is a lot of mirrors where you can download any program from NetKit, I just downloaded [bsd-finger-0.17.tar.gz] and after extracting the source a grepped it for lstat()

```
giggles@ctrl-c:~/lab$ grep -sr lstat bsd-finger-0.17
bsd-finger-0.17/finger/lprint.c:        if (lstat(tbuf, &sbuf1) ||
!S_ISREG(sbuf1.st_mode)) return 0;
```

OH nice!, We spotted where and why our fifo is being ignored, we cant just use a fifo as our .plan file because its being checked! Never imagined someone will really check if the file is regular... I will also reproduce the entire function here

```
static int
show_text(const char *directory, const char *file_name, const char *header)
{
        int ch, lastc = 0, fd;
        FILE *fp;
        struct stat sbuf1, sbuf2;

        snprintf(tbuf, TBUFLEN, "%s/%s", directory, file_name);

        if (lstat(tbuf, &sbuf1) || !S_ISREG(sbuf1.st_mode)) return 0;
        fd = open(tbuf, O_RDONLY);
        if (fd<0) return 0;
        if (fstat(fd, &sbuf2)) { close(fd); return 0; }
        /* if we didn't get the same file both times, bail */
        if (sbuf1.st_dev!=sbuf2.st_dev || sbuf1.st_ino!=sbuf2.st_ino) {
                close(fd);
                return 0;
          }
        fp = fdopen(fd, "r");
        if (fp == NULL) { close(fd); return 0; }


         xprintf("%s", header);
        while ((ch = getc(fp)) != EOF) {
                xputc(ch);
                lastc = ch;
        }
        if (lastc != '\n') xputc('\n');

        fclose(fp);
        return 1;
}
```

The first idea that I have to give me time to write dynamic content was trying to fool finger making use of race condition between the the lstat() and open() but unfortunately the developer of finger is clever and he check the st_dev and st_ino fields of the opened file to make sure lstated file and opened file are the same, no luck on this idea it seems that its really impossible to make finger to reach the while loop if our file(.plan) is not created as a regular file.

Thinking more about this I still want finger to open my lovely fifo, and maybe there is way, if linux reuse inode number when you remove and create a file in sequence, then we have a chance to fool all conditions tests the show_test() function, to validate this I just use a one-liner to create .plan as regular file, check the inode, remove .plan and create it as a fifo and check the inode again, I also created a useless file between the tests to make sure it really works as expected, here is the results:

```
giggles@ctrl-c:~$ touch .plan; ls -il .plan; rm .plan; mkfifo .plan; ls -il
.plan; rm .plan
612397 -rw-rw-r-- 1 giggles giggles 0 Jul  2 06:13 .plan
612397 prw-rw-r-- 1 giggles giggles 0 Jul  2 06:13 .plan
giggles@ctrl-c:~$ touch k
giggles@ctrl-c:~$ touch .plan; ls -il .plan; rm .plan; mkfifo .plan; ls -il
.plan; rm .plan
612399 -rw-rw-r-- 1 giggles giggles 0 Jul  2 06:13 .plan
612399 prw-rw-r-- 1 giggles giggles 0 Jul  2 06:13 .plan
giggles@ctrl-c:~$ touch .plan; ls -il .plan; rm .plan; mkfifo .plan; ls -il
.plan; rm .plan
612400 -rw-rw-r-- 1 giggles giggles 0 Jul  2 06:13 .plan
612400 prw-rw-r-- 1 giggles giggles 0 Jul  2 06:13 .plan
```

That’s good for us, if we can unlink the file and recreate it fast as sonic we can fool finger, this is just for fun since it’s not possible to rely in race condition because it will be very hard to make the things happen in the correct order, but as a simple proof of concept I create the following program race.c

which create .plan and replace it with a fifo and then write a message if some other process had open the fifo to read (this process should be finger in our case):

```
#include
#include
#include
#include
#include

int main(int argc, char **argv) {
again:
        int fd = creat(".plan", S_IRWXU|S_IRWXO|S_IRWXG);
        close(fd);
        unlink(".plan");
        mkfifo(".plan", S_IRWXU|S_IRWXO|S_IRWXG);
        int fifo = open(".plan", O_WRONLY|O_NONBLOCK);
        if (fifo>0) {
                write(fifo, "We love ^C\n", 11);
                close(fifo);
                goto end;
        }
        unlink(".plan");
goto again;
end:
        printf("We won!\n");
        return 0;
}
```

This program loop until it successfully manage to win the race condition, I ran it on tmux using strace until the victory, and for this I needed to run finger in a infinity loop too, here is the results, in the top is the race program and down is the one-liner running finger

```
giggles@ctrl-c:~$ strace ./a.out
...
creat(".plan", 0777)                    = 3
close(3)                                = 0
unlink(".plan")                         = 0
mknodat(AT_FDCWD, ".plan", S_IFIFO|0777) = 0
```

```
openat(AT_FDCWD, ".plan", O_WRONLY|O_NONBLOCK) = -1 ENXIO (No such device or
address)
unlink(".plan")                         = 0
creat(".plan", 0777)                    = 3
close(3)                                = 0
unlink(".plan")                         = 0
mknodat(AT_FDCWD, ".plan", S_IFIFO|0777) = 0
openat(AT_FDCWD, ".plan", O_WRONLY|O_NONBLOCK) = 3
write(3, "We love ^C\n", 11)            = 11
close(3)                                = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x70), ...},
AT_EMPTY_PATH) = 0
getrandom("\x28\xce\xa5\x06\x6a\x9c\x9e\x3f", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55b371c46000
brk(0x55b371c67000)                     = 0x55b371c67000
write(1, "We won!\n", 8We won!
```

```
giggles@ctrl-c:~$ while true; do  finger giggles | grep "\^"; done
We love ^C
```

What happened here was that finger used lstat and "confirmed" that .plan is a regular file, but before it calls open() our program removed .plan and created a fifo with the same name, fast enough to reuse the same inode number, after this all the checks go fine and finger did not know it opened a fifo, since it opens it readonly mode, our race program will finally manage to open the writing end of fifo and send the content to finger through the fifo! As you can see it needs to happen in a very specific order.

I know that this dont make possible to make finger content dynamic in a deterministic manner, but exploiting this race condition to bypass the checks and showing that finger can read content from a non regular file is cool enough to be worth to add it all here, even that we don’t reached our goal yet, we found a little bug in finger :)

This is the end of this article, I tried to research a bit on how can I lock a file on linux, but all ways to do this is not reliable and need to cooperation of other process, by default its not possible to make a regular file to block on read so we have time to add content. Well maybe someone know a way to make open() or read() to block on a regular file, I got really curious about this, if you have an idea let me know ^.^

A Smol Interview with ~calamitous (^C admin) conducted by ~loghead (^Z editor/compiler)

What made you want to start a pubnix?

In college in the early 90s, I ran a BBS, and I was quite fond of the little community that grew there. It went away once I got on the internet, and unfortunately, I managed to almost completely miss the advent of pubnixes, though I dabbled a little with my university shell and some MUDs.

I was bitten by the "tilde" bug when Paul Ford founded Tilde.club (http://tilde.club) in 2014. I was a little late to the game, and by the time I tried to sign up, the waiting list was already 1,200 requests deep. So I decided that there was enough interest for me to start my own server in a similar vein.

I'm a programmer that loves fiddling with new bleeding-egde and esoteric programming languages. I enjoy exploring new concepts and paradigms in programming. The problem is that setting up a fresh language-- especially one that hasn't been around long enough to get some easy installers and configuration-- can be difficult and time-consuming. That's a large part of what inspired me to start Ctrl-C.club, to make some of those languages available to everybody, without the hassle of setting everything up.

Ctrl-C.club was officially started in December 2014, making it one of the longest continuously-running tildes, as far as I know.

Naturally, over the years, we've expanded into lots of other areas (websites, gemini capsules, games, irc, etc.), but we still offer an array of about 30 programming languages. This list continues to grow.

What are your thoughts on decentralization?

Hoo, now that's quite a topic. :) I'd like to lump my answer for this in with the next question, since they're so closely related, in my opinion.

Do you think the "Smol Web" is here? Will Mastodon, Gopher, Gemini and other protocols be the predominant services people use for their presence online?

On the one hand, I don't think that "smol" services will ever be the predominant services-- I believe the barriers to entry, friction, and technical skill required are too high for the majority of internet users today.

That said, I don't think that the "smol web" was ever _not_ here. The early internet was driven by hobbyists and technologists, playing with this new medium and seeing how they could stretch the boundaries of what was possible.

As time went on and the technology grew, the process of publishing to the web became easier and more democratic-- no longer did you need an expensive T1 line directly to your home or years of experience as a sysadmin or programmer to publish on the web.

With the resulting explosion of content creators, centralization increased. It's easier and more economical to host and administer 100 sites on a single server than on 100 small,

scattered servers. Centralization was, and is, an economic inevitability.

Centralization is not in and of itself a bad thing-- witness the democratization of the web-- but concentrating control in too few hands has led to pathological failure modes-- censorship, control, and some profoundly alarming behaviors around surveillance and invasion of privacy.

Any source of power and control will attract those whose purpose is to gain and maintain power at any cost. This is the iron-clad law of human nature. As Lord Acton stated so eloquently, "Power tends to corrupt and absolute power corrupts absolutely."

In my opinion, the only true antidote is to reduce and disperse power. In terms of the internet, decentralization solves this nicely-- it's easier and more economical to control 100 people who are communicating through approved channels on your server than 100 people scattered across a mishmash of services: publishing, speaking, and building as they like.

There are serious downsides to decentralization, of course. It's much more expensive, it's more difficult to find and promote interesting voices, has higher requirements for technical expertise, and provides more nooks for extreme and unsavory content to hide and grow. But it's clear to me that these detriments, however difficult or distasteful they might be, are far preferable to a sanitized, managed, and "safe" internet.

If you could "shout out" five resources, tools, blogs, services online that people don't know about, but should know about, what would they be?

I would not presume to know many hidden "must-have" resources, but I can certainly list off a few of my favorites, especially as they relate to being a programmer:

1. Tool: Vim. (https://www.vim.org/) It's powerful, it's fast, and it's ubiquitous. It's also got a learning curve like a brick wall. I have an hour-long presentation I give on the hows and whys of Vim, but in the interests of not turning this into a novel-length love letter, I'll just say it's one of those rare pieces of software which rewards mastery over and above the admittedly significant cost of learning it well.

2. Site: Hacker News (https://news.ycombinator.com/) Though the culture has suffered a little lately from the Reddit diaspora landing on them, this is still a wonderful resource for the curious and for technologists to find interesting and amazing technologies to learn about.

3. Tool: Git. (https://git-scm.com/) (Not Github which is now centralized under Microsoft umbrella). Git is a wonderful tool that builds a narrow bridge across the gap between abstract CS concepts and real, useful software. Most programmers use it, but very few understand it deeply. I would strongly encourage every programmer to learn all its options, and the many ways it can be used. I would also encourage anyone who uses it to learn how it structures and stores data. It's elegant and almost breathtakingly simple at its core.

4. Language: Ruby (https://www.ruby-lang.org/) There are hundreds of programming languages across dozens of paradigms, and most of them are fun to use, but my personal favorite is Ruby. It is simple, comprehensible, composable, and elegant. It is also one of the slowest languages around, but writing (and reading!) code in Ruby is such a joy that it's usually the first tool I reach for.

5. Mindset: Simplicity. Our world is filled with complexity, and we're wrapping more around it all the time. There's a constant pressure to add complexity: not only to our code, but to our

jobs, and even to our lives. Simplifying-- the act of stepping back and asking "what can I take away?"-- is difficult, but also incredibly liberating. It's a very broad topic, but the more places in my life that I can aggressively shave away the fluff and unnecessary complexity, the more I can understand and focus on what's truly important to me.

6. BONUS: Service: Ctrl-C.club! (https://ctrl-c.club/) Can I just plug my own service? :D There are many great tildes and pubnixes, but in my totally unbiased and completely objective opinion, Ctrl-C.club has the best users with the most interesting projects going. ;)

Of course, there are many thousands of other amazing resources, services, and tools, and I would not insult anybody's intelligence by pretending that this list represents the very best. It's just a few things I like. :)

What is the future of ^C?

More of the same! I don't have any grand aspirations to turn Ctrl-C.club into an internet destination, or a megasite, or a profitable venture. It's a small, nice community of learners and builders, and my goal is to keep it that way. :) I want to make sure we always have a place to create and share the great stuff we're working on, whether it's sites, software, zines, poetry, or whatever else.

Of course, I plan to keep us up with new developments in the "smol web"-- probably the biggest "smol" technology to come out in the last few years was Gemini, the simpler, privacy-protecting protocol for sites. This technology was actually recommended by our users, and has become one of our most popular features, as it fits so well with our culture.

I'm happy to see everybody learning, building and having fun. Ctrl-C.club has ended up being better than I'd ever expected, and I'm excited to see what we do in the coming years.

PHEW! Let’s wrap it there, all! 30 pages of goodness that I hope all take time to absorb and enjoy! :) The details on the ISSN for ^Z are still in the works, the status is “moving along”, but not there yet. In time.

Thank you to everyone who reads Ctrl-ZINE. All contributors and collaborators. People who share, download, print, and take benefit from the content within its pages. Compiling each issue is a joy, and I am forever thankful for all who add-in, as well as those getting a copy to learn-from!

Everyone stay well, and we’ll see you in Issue 6!

~loghead