Diving into HDFS

in strace

Yesterday I wanted to start learning about how HDFS (the Hadoop Distributed File System) works internally. I knew that

- It's distributed, so one file may be stored across many different machines
- There's a *namenode*, which keeps track of where all the files are stored
- There are *data nodes*, which contain the actual file data

But I wasn't quite sure how to get started! I knew how to navigate the filesystem from the command line (`hadoop fs -ls /`, and friends), but not how to figure out how it works internally.

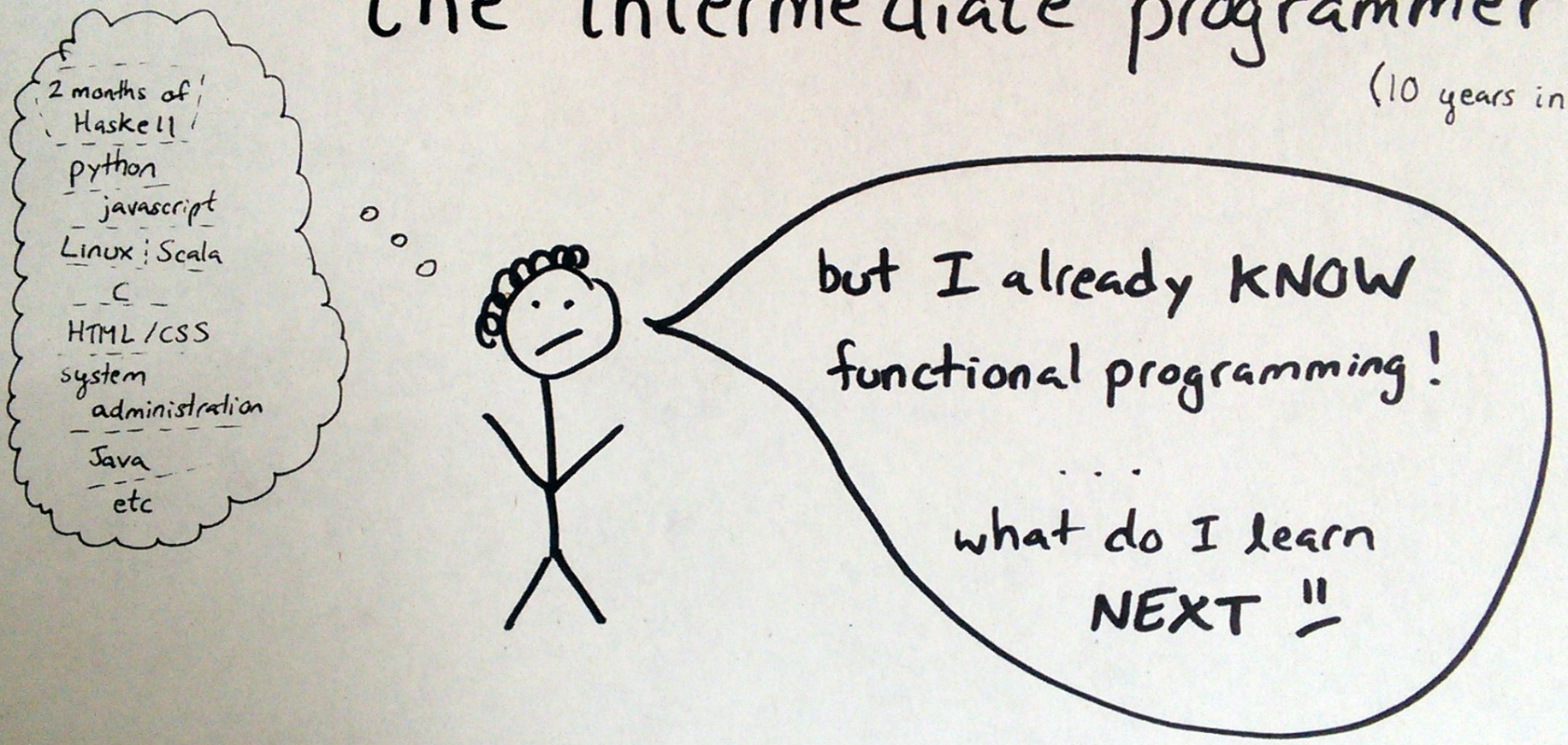
SYSTEMS PROGRAMMING IS FOR EVERYONE

Julia Evans
Stripe

- twitter: @b0rk
- blog: jvns.ca

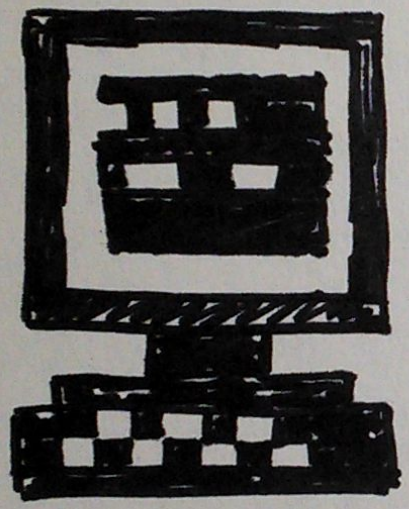
the intermediate programmer

(10 years in)



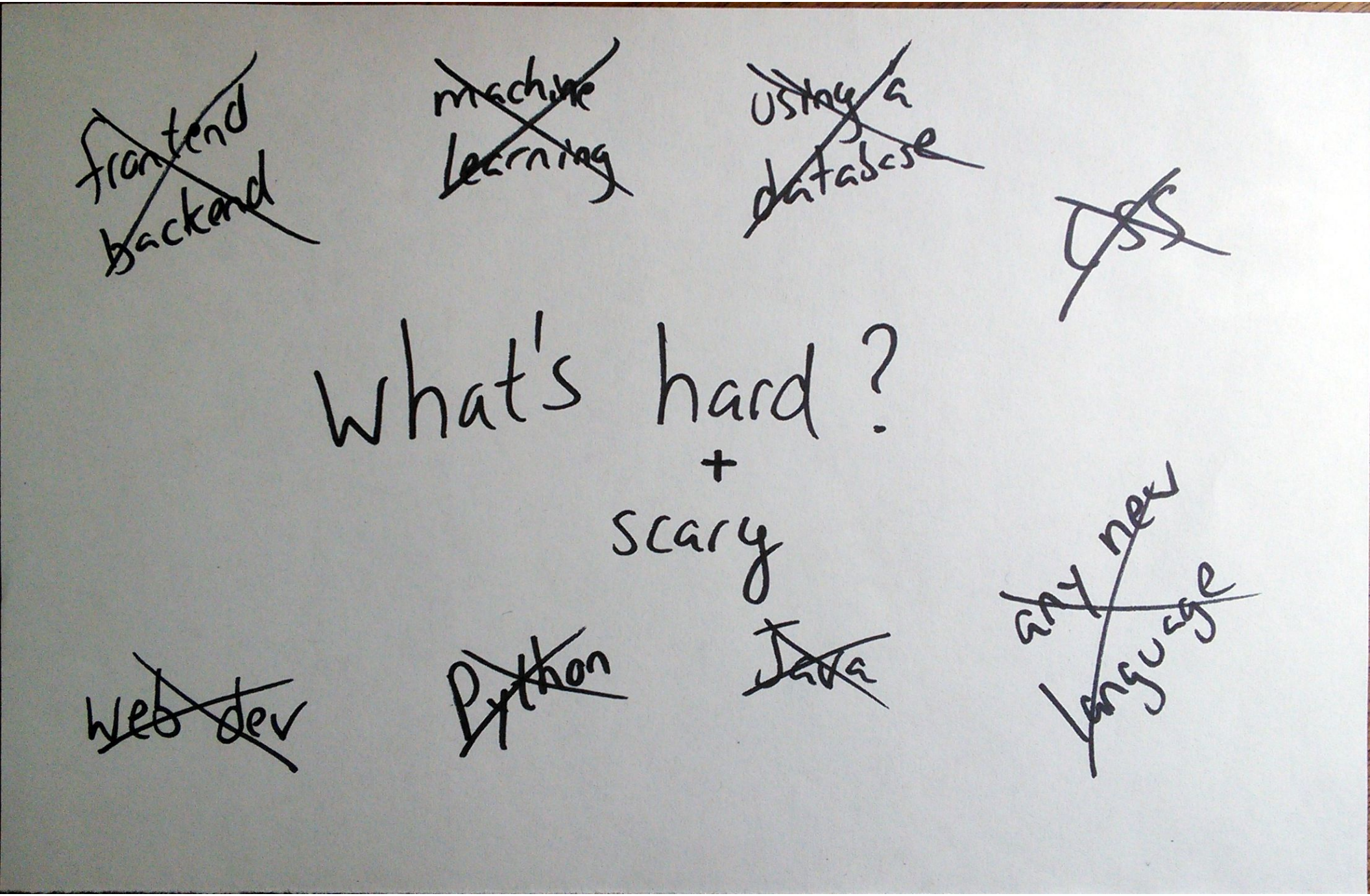
the Recurse Center

New York!
12 weeks!
free!



the best
programming
community
in the world

recurse.com


frontend
backend
machine learning
using a database
CSS
what's hard?
+
scary
web dev
Python
Java
any new language

fun with debugging

...

programming experiments

debugging

perl | go | c++ | fortran
php | python | java
golang | smalltalk
INTERCAL | BASIC

LINUX-ONLY

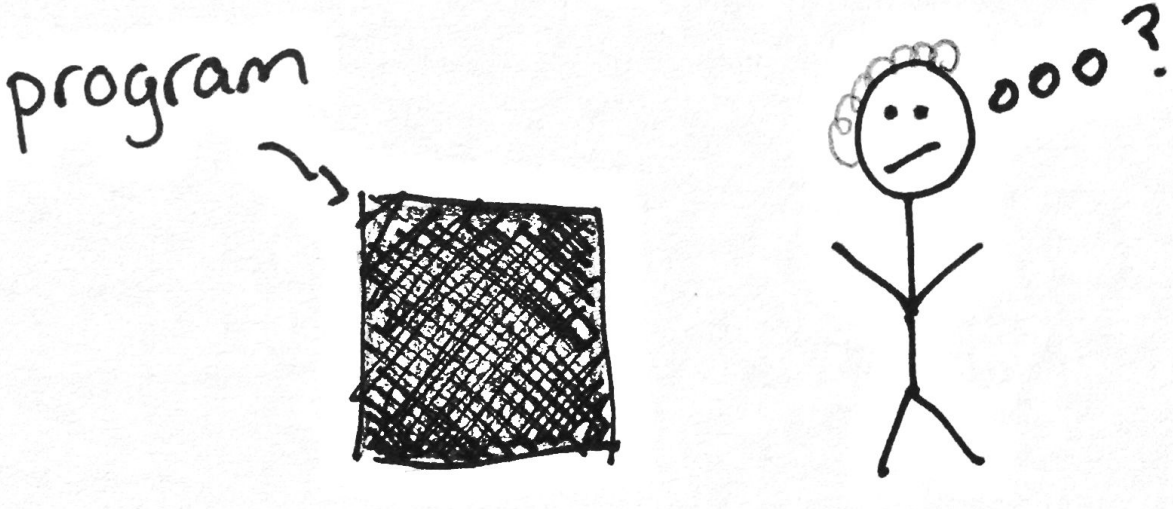
DEBUGGING:

- look at the source code
- add print statements
- know the programming language

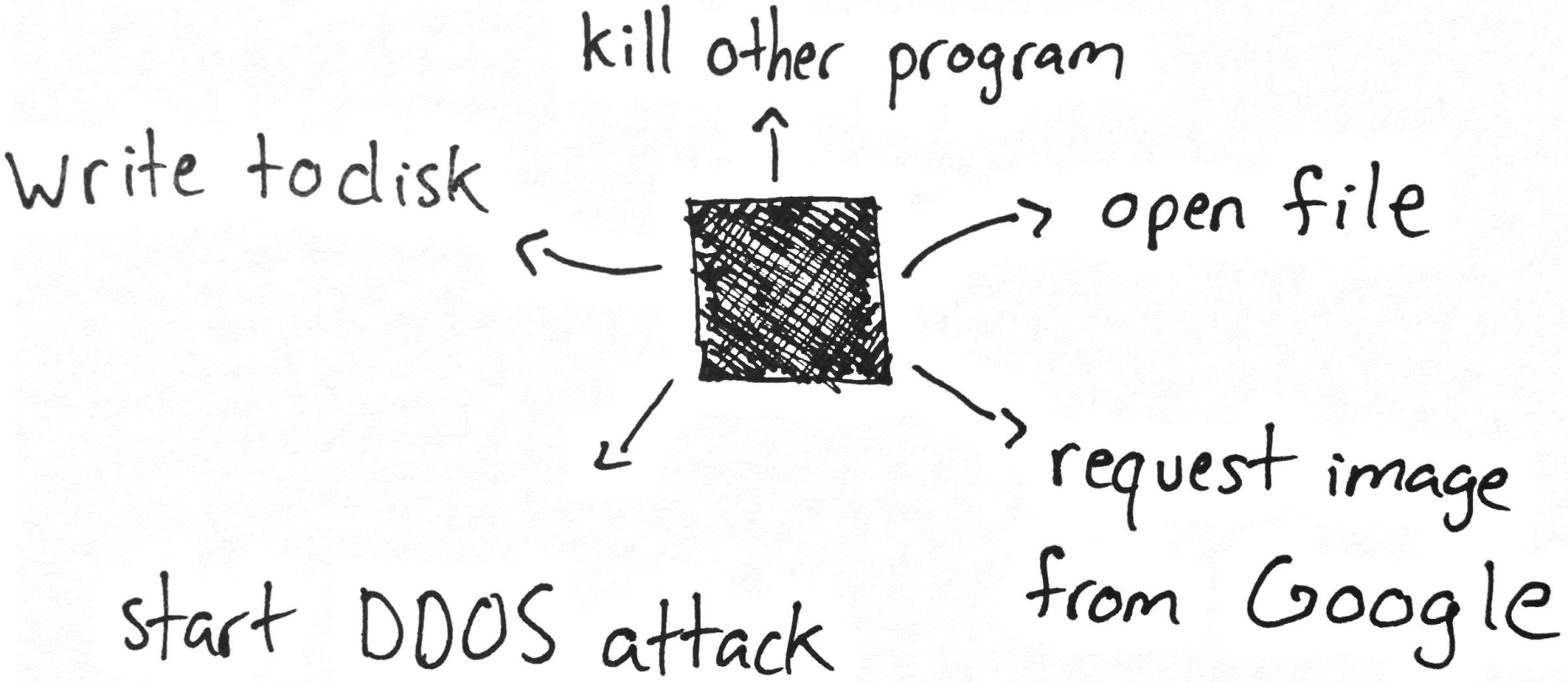
DEBUGGING:

- ~~look at the source code~~
- ~~add print statements~~
- ~~know the programming language~~
- ★★★ be a wizard★★★

BLACK BOX TOOLS



outside the box



inside the box

call function

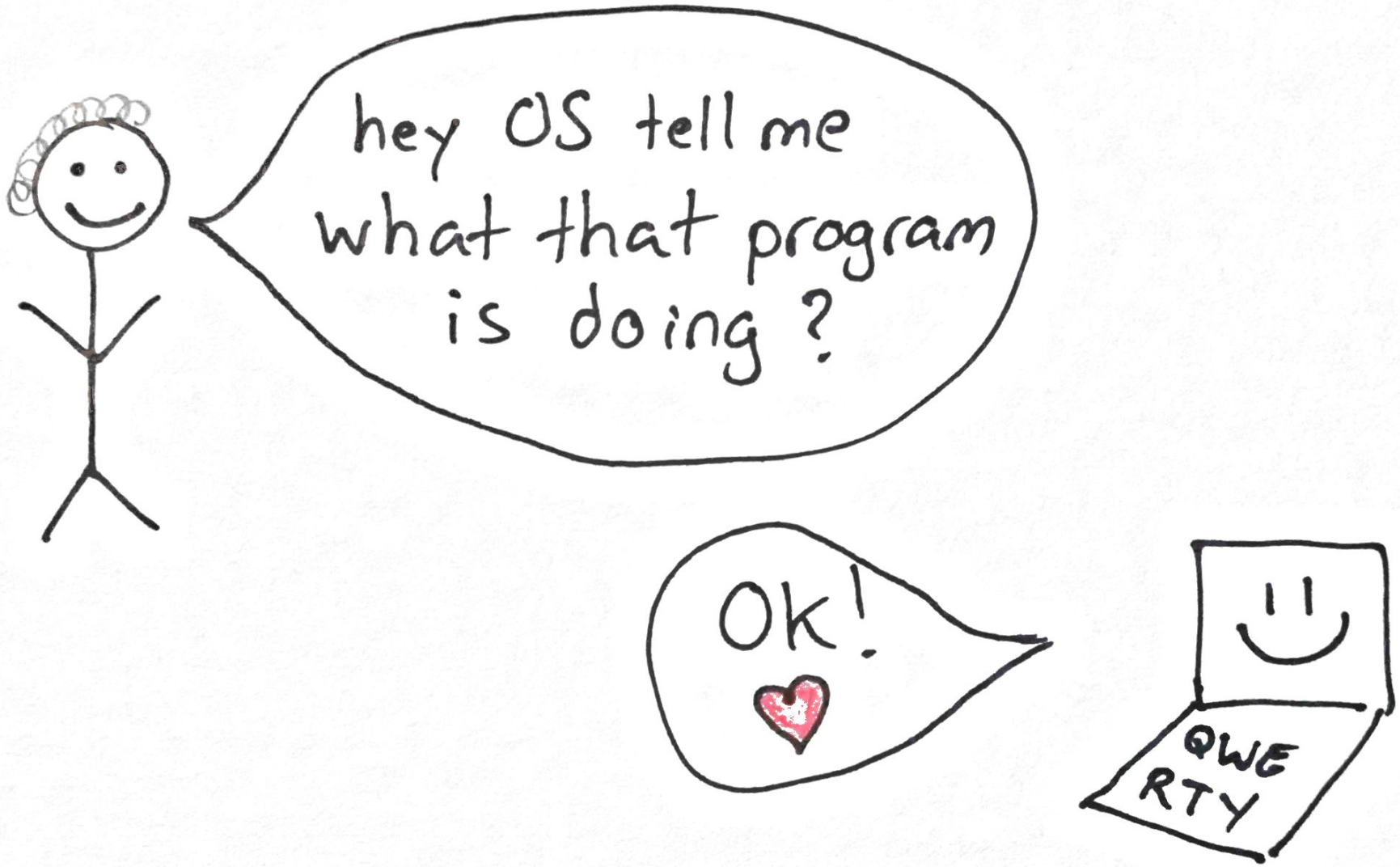
access memory

do a loop

set variable

calculate something

black box tools:



```
top - 22:00:30 up 1 day, 21:54,  6 users,  load average: 0.54, 0.63, 0.78
Tasks: 354 total,   3 running, 345 sleeping,   0 stopped,   6 zombie
Cpu(s):  9.4%us,  3.7%sy,  0.0%ni, 86.5%id,  0.4%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   7869048k total,  7513520k used,   355528k free,   385404k buffers
Swap:  7811068k total,   217852k used,  7593216k free,  2928032k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
12520 root      20   0  277m  34m  21m S   10  0.5  13:14.42 Xorg
13045 bork      20   0 1842m 273m  57m S    9  3.6  43:14.55 chrome
31500 bork      20   0  873m 183m  19m S    8  2.4  19:49.65 chrome
23767 bork      20   0 1607m  95m  27m S    6  1.2   7:39.97 compiz
20927 bork      20   0 1509m 422m  25m S    3  5.5  27:15.33 chrome
 1741 bork      20   0  508m  16m  12m S    3  0.2   0:00.35 gnome-screensho
31509 bork      20   0 84.1g  66m  11m S    3  0.9  17:14.32 nacl_helper
13124 bork      20   0  906m 187m  29m S    3  2.4  12:44.65 chrome
31484 bork      20   0  940m 219m  27m S    2  2.9   7:46.46 chrome
12895 bork       9 -11  418m 6108 3664 S    2  0.1   4:58.72 pulseaudio
    7 root      20   0     0    0    0 S    0  0.0   0:27.43 rcu_sched
   10 root      20   0     0    0    0 S    0  0.0   0:13.18 rcuos/2
  722 root      20   0     0    0    0 R    0  0.0   0:01.17 kworker/0:1
 1102 root      20   0     0    0    0 R    0  0.0   0:00.39 kworker/2:2
 1736 bork      20   0 17592 1668 1092 R    0  0.0   0:00.10 top
 2078 postgres  20   0  128m 1768  504 S    0  0.0   0:08.12 postgres
12800 bork      20   0  899m  24m  12m S    0  0.3   0:18.88 gnome-settings-
12920 bork      20   0  399m  14m  10m S    0  0.2   0:09.00 gtk-window-deco
13172 bork      20   0  704m  40m  17m S    0  0.5   1:45.79 chrome
13466 bork      20   0  335m  19m  16m S    0  0.3   0:04.22 gnome-screensav
16911 bork      20   0  590m  19m  11m S    0  0.3   1:01.41 gnome-terminal
```

THIS TALK

- Wizard school (or, an operating systems primer)
- Chapter 1: The Case of the Mystery Config File
- Chapter 2: The Case of the French Website
- Chapter 3: The Case of the Slow Program

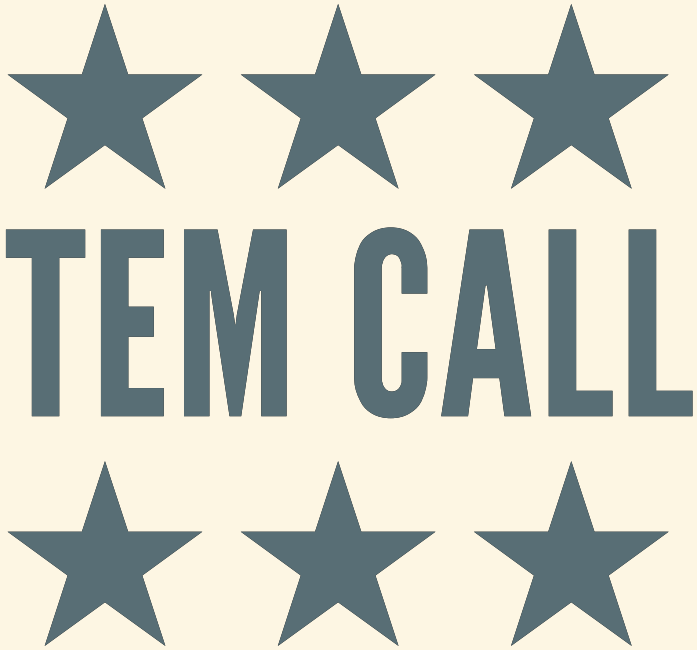
WIZARD SCHOOL
-OR-
WHY YOU SHOULD ❤ YOUR OPERATING SYS-TEM

WHAT IS AN OPERATING SYSTEM FOR?

When I go to http://google.com, kernel code runs for:

- Typing in the address
- Handling every network packet
- Writing history files to disk
- Allocating memory
- Communicating with the graphics card

HOW TO CALL OPERATING SYSTEM CODE


SYSTEM CALLS!!!

SYSTEM CALLS: AN OS'S INTERFACE

- open a file! (`open`)
- start a program! (`execve`)
- change a file's permissions! (`chmod`)

WHAT WE'VE LEARNED

- Your OS does tons of stuff
- Programs tell it what to do using system calls

USING SYSTEMS KNOWLEDGE TO DEBUG

CHAPTER 1: THE CASE OF THE MYSTERY CONFIG FILE

Does bash use `.bash_profile` or `.bashrc`??!??

STRACE
=
WIZARDRY

STRACE

=

TRACING SYSTEM CALLS

HOW TO STRACE

```
$ strace google-chrome
execve("/usr/bin/google-chrome", ["google-chrome"], [/* 51 vars */])
brk(0)                                  = 0x124f000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
```

OPEN

strace -e open bash

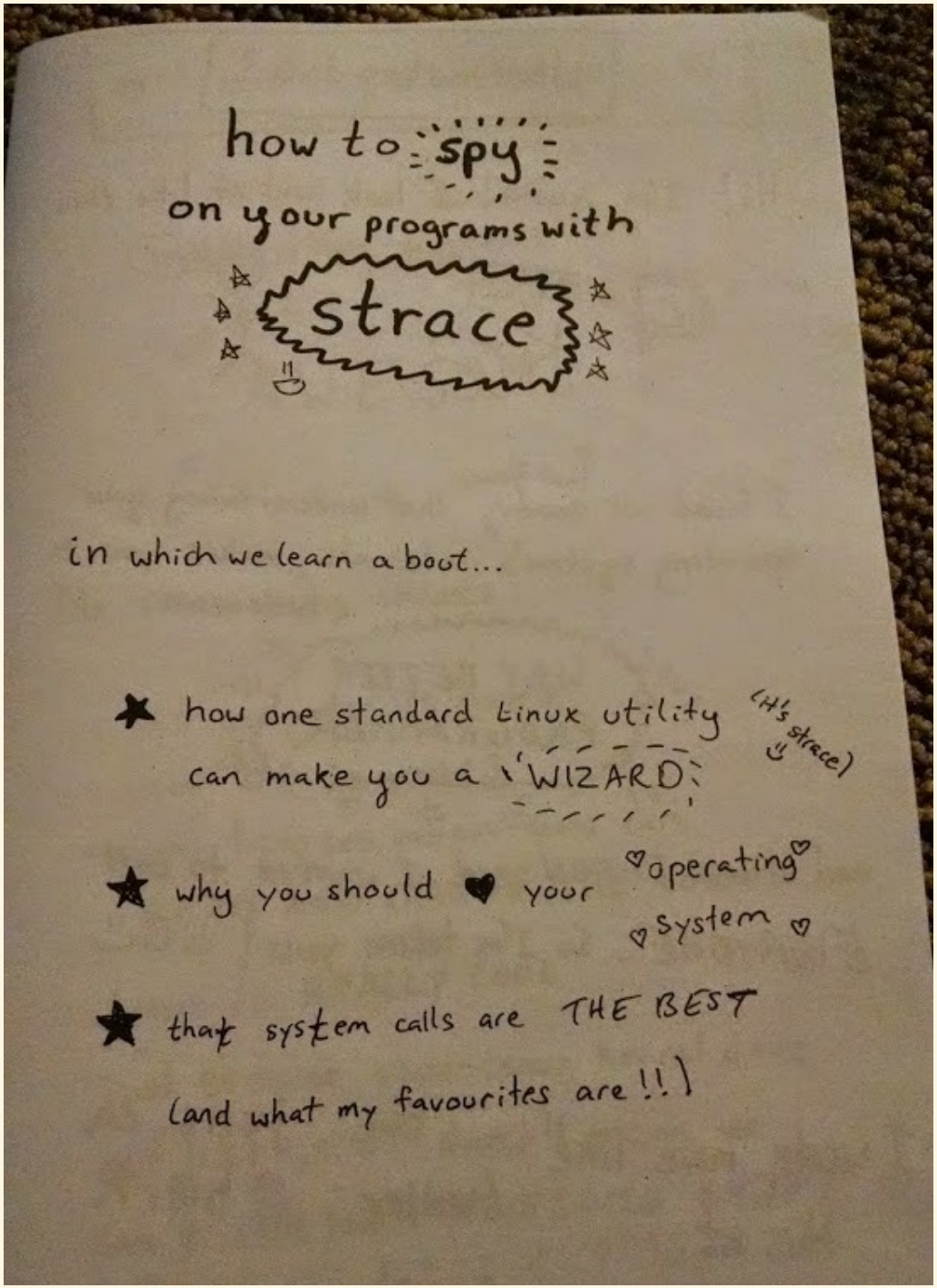
```
8421  open("/home/bork/.bashrc", O_RDONLY) = 3
8421  open("/home/bork/bin/google-cloud-sdk/path.bash.inc", O_RDONLY) = 3
8421  open("/home/bork/bin/google-cloud-sdk/completion.bash.inc", O_RDONLY) = 3
8421  open("/home/bork/.bash_history", O_RDONLY) = 3
```

BASHRC WINS!

OTHER AWESOME SYSTEM CALLS

- `write` for log files
- `execve` for starting programs
- `recvfrom` for receiving data

STRACE ZINE

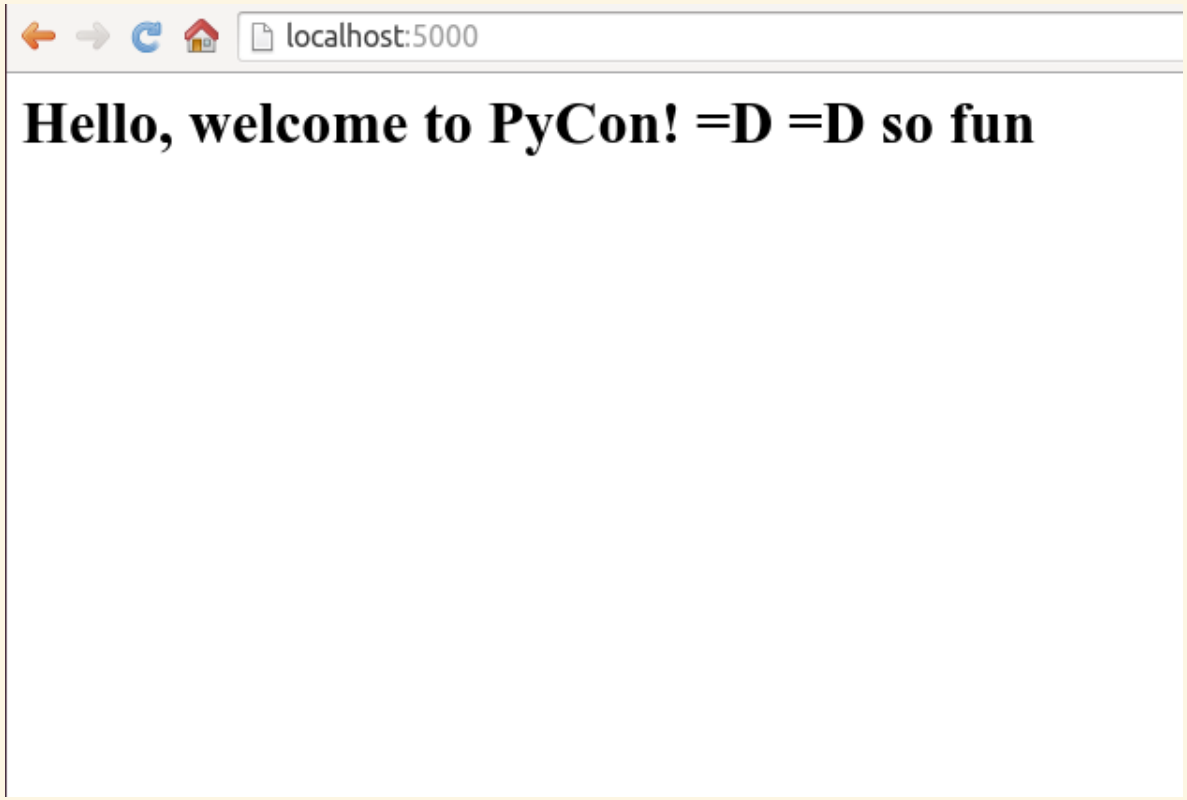

how to spy
on your programs with
strace
in which we learn a bout...
★ how one standard Linux utility can make you a WIZARD (it's strace)
★ why you should ♥ your ♡operating♡ ♡system♡
★ that system calls are THE BEST (and what my favourites are!!)

way 2: eBPF tools

https://github.com/iovisor/bcc-tools

opensnoop
&
execsnoop

CHAPTER 2: THE CASE OF THE FRENCH WEBSITE


localhost:5000
Hello, welcome to PyCon! =D =D so fun

```
bork@kiwi~> curl localhost:5000

    <html>
    <body>
    <h1>Bonjour, bienvenue à PyCon!</h1>
    </body>
    </html>
    ⏎
```

???

NETWORK SPYING TO THE RESCUE

```
sudo ngrep -d lo 5000
interface: lo (127.0.0.0/255.0.0.0)
match: 5000
####
T 127.0.0.1:45438 -> 127.0.0.1:5000 [AP]
  GET / HTTP/1.1..Host: localhost:5000..Connection:
keep-alive..Cache-Control: max-age=0..Accept:
text/html,application/xhtml+xml,application
/xml;q=0.9,image/webp,*/*;q=0.8..User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.53 Sa
ari/537.36..DNT: 1..Accept-Encoding: gzip, deflate,
sdch..Accept-Language: en-US,en;q=0.8..Cookie:
username-localhost-8888="2|1:0|10:142841
1879|23:username-localhost-8888|48:MjYzMTc2NGMtYTA1MC00YjNkLTkyYTktNG
fab7ee279"....
##########################
T 127.0.0.1:45440 -> 127.0.0.1:5000 [AP]
  GET / HTTP/1.1..User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu)
libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
librtmp/2.3..Host: localhost:5000..Accept: */*....
####################
```

Accept-Language: en-US

```
bork@kiwi~> curl --header "Accept-Language: en-US" localhost:5000

    <html>
    <body>
    <h1>Hello, welcome to PyCon! =D =D so fun</h1>
    </body>
    </html>
    ⏎
```

NETWORK SPYING TOOLS

- ngrep
- tcpdump
- wireshark
- mitmproxy

CHAPTER 3: THE CASE OF THE SLOW PROGRAM

3 SLOW PROGRAMS

1. CPU time
2. too many writes
3. waiting for a slow server

MYSTERY PROGRAM #1

```
$ time python mystery_1.py
0.09user 0.01system 0:02.11elapsed 5%C
```

WHAT IS IT WAITING FOR?

LET'S LOOK INTO THE KERNEL'S SOUL

/PROC/PID/STACK

```
$ pgrep -f mystery_1
31728
$ sudo cat /proc/31728/stack
[<ffffffff81702467>] sk_wait_data+0x107/0x120
[<ffffffff81767112>] tcp_recvmsg+0x2e2/0xb80
[<ffffffff81794d6e>] inet_recvmsg+0x7e/0xb0
[<ffffffff816fdb6b>] sock_recvmsg+0x3b/0x50
[<ffffffff816fddc1>] SYSC_recvfrom+0xe1/0x160
[<ffffffff816ff1ce>] SyS_recvfrom+0xe/0x10
[<ffffffff818244f2>] entry_SYSCALL_64_fastpath+0x16/0x71
[<ffffffffffffffff>] 0xffffffffffffffff
```

WE WIN! IT WAS THE NETWORK!

OUR SERVER

```
@app.route('/')
def slow():
    time.sleep(2)
    return "Hi!"
app.run()
```

MYSTERY PROGRAM #2

```
$ time python mystery_2.py
2.74user 0.00system 0:02.74elapsed 99%
```

USE A PYTHON PROFILER

```
total = 0
for i in xrange(14000000):
    total += i
```

MYSTERY PROGRAM #3

(REALLY A MYSTERY)

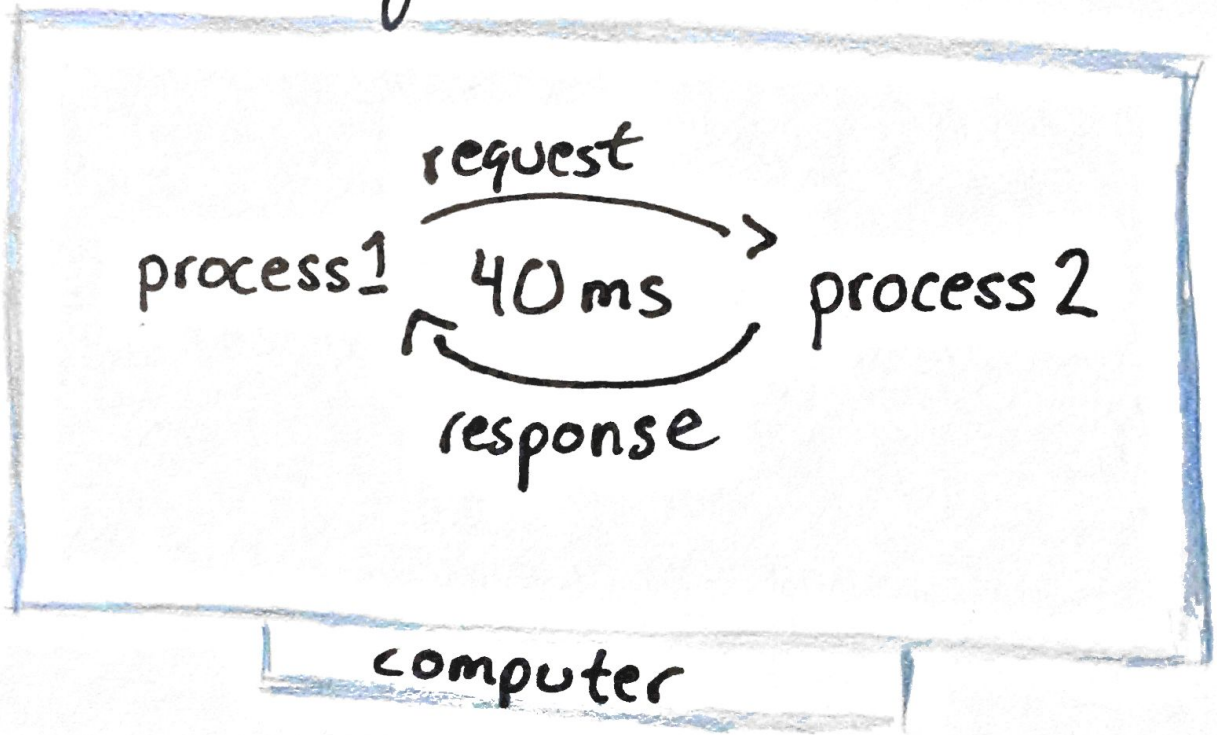
```
$ time python mystery_3.py
0:02.61elapsed 62%CPU
$ time python mystery_3.py
0:10.61elapsed 10%CPU
```

DEMO DEMO

WE WIN

a hard problem

storytime!



40 milliseconds inside the same computer is

goal: 1000x faster

step 1

google

smart colleagues

print statements

step 2: more information!

less

program 1: hi!

<40 ms passes>

program 2: oh hi!

more

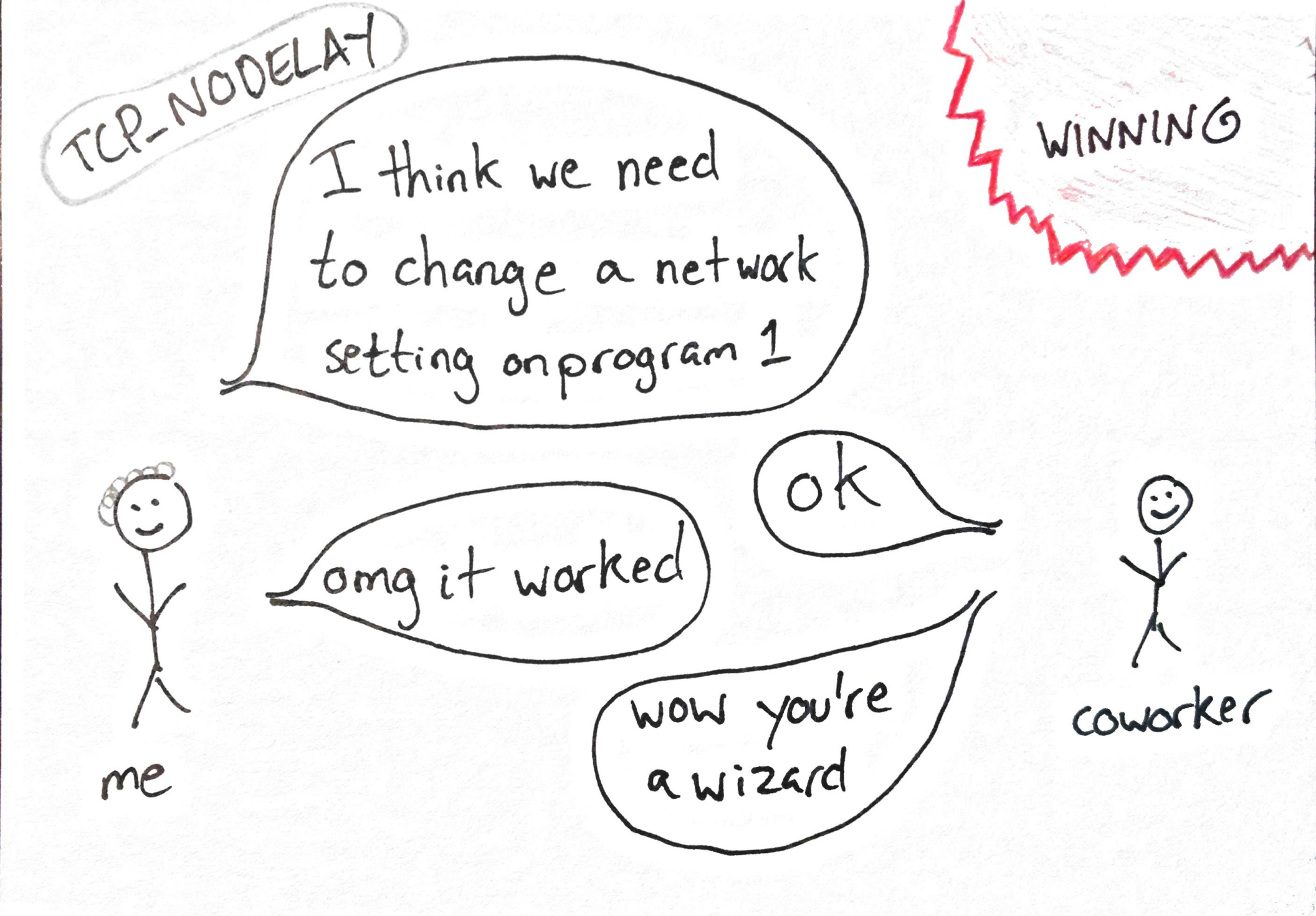
program 1: <packet 1>

<40 ms passes>

program 1: <packet 2>

program 2: oh hi!

blame program 1


TCP_NODELAY
I think we need to change a network setting on program 1
WINNING
ok
omg it worked
wow you're a wizard
me
coworker

tcpdump is amazing

YOUR PROGRAM

=

BLACK BOX

THERE ARE A LOT OF AWESOME TOOLS

I'm not

the hero

the tools

are the heroes

strace

wireshark

dstat

lsof

LEARN YOUR OPERATING SYSTEM

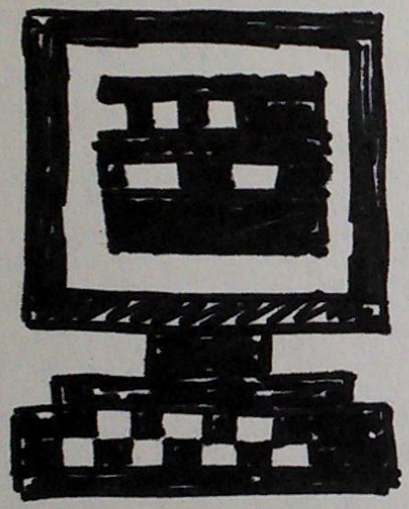
fun with debugging

...

programming experiments

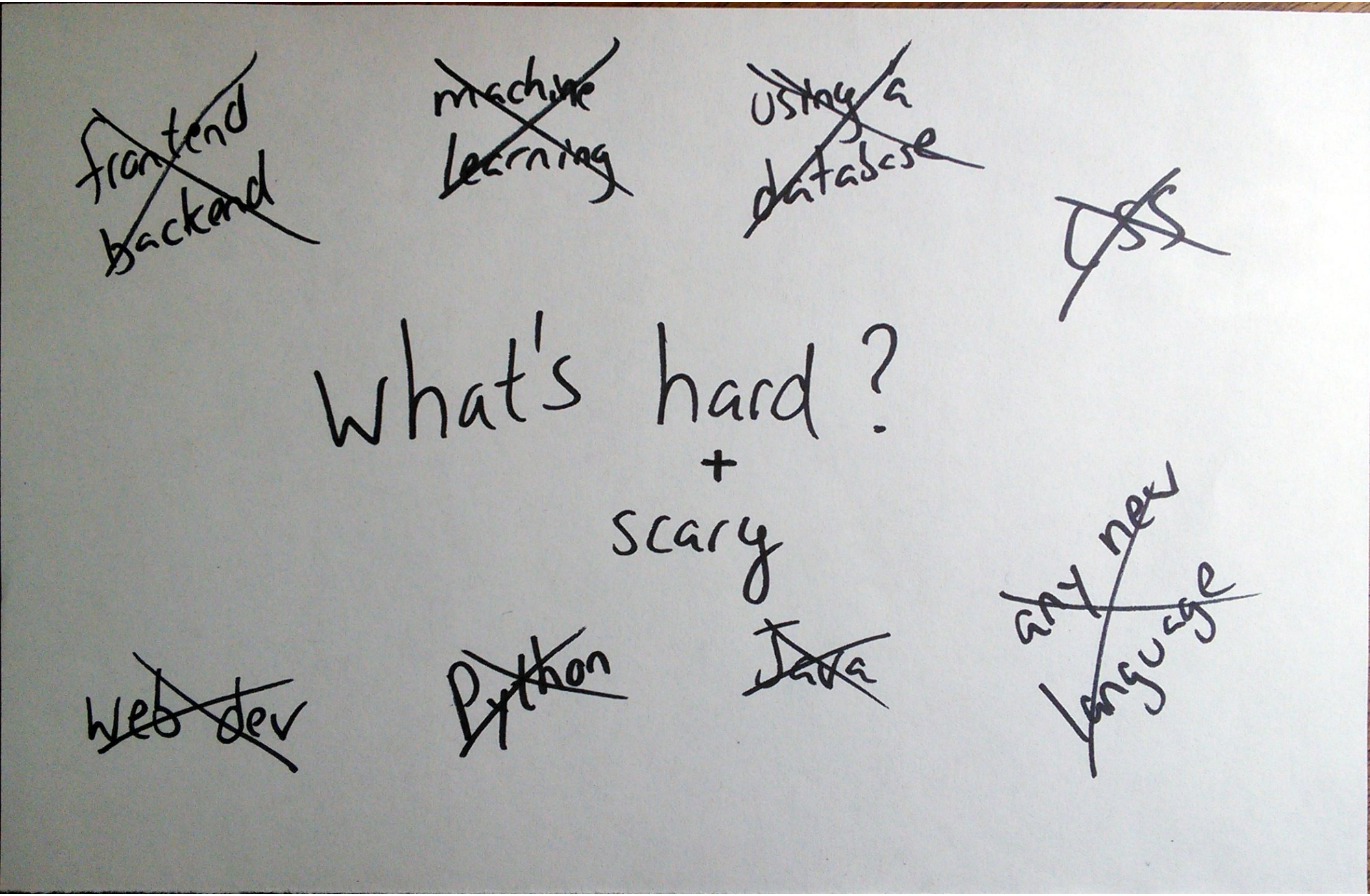
the Recurse Center

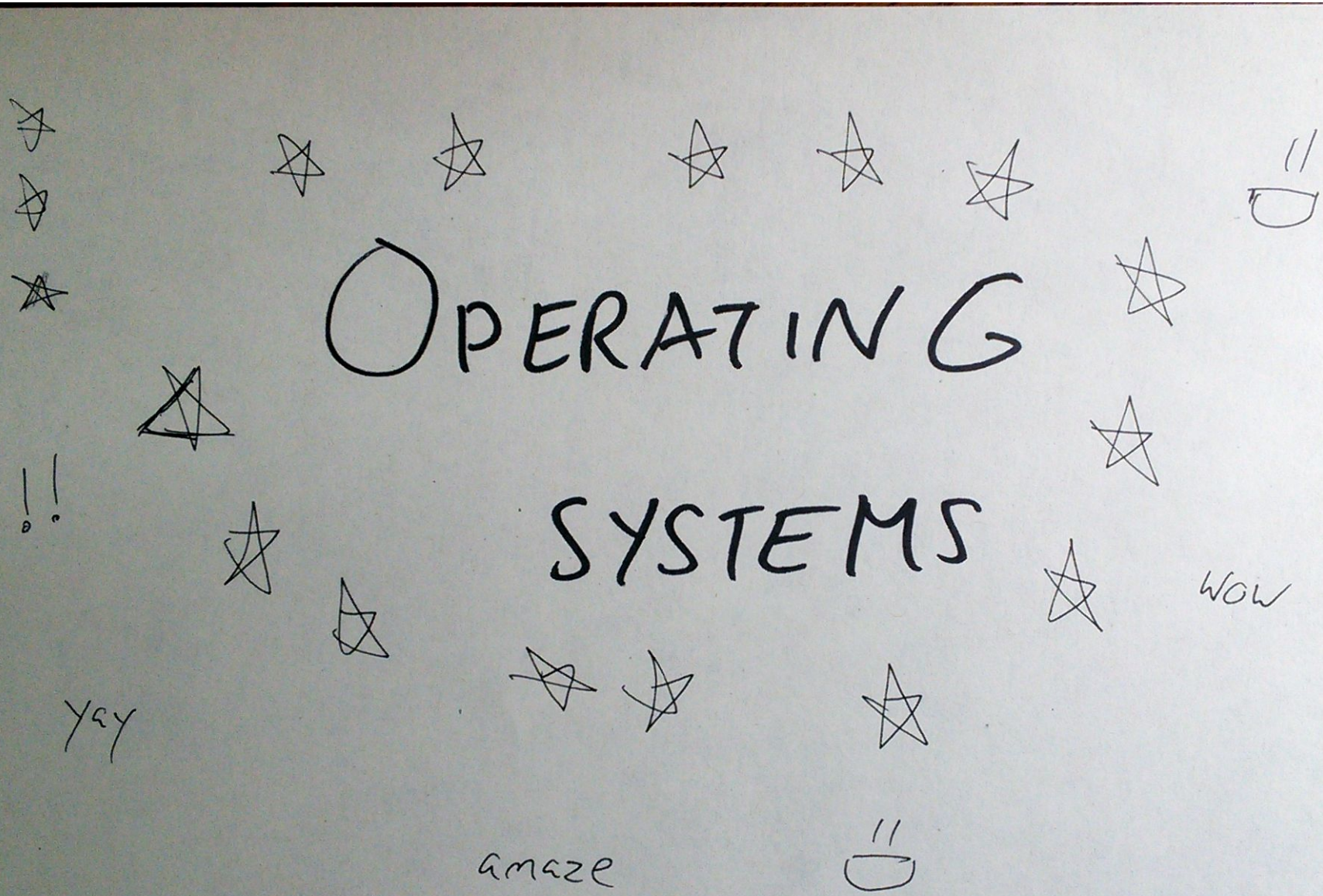
New York!
12 weeks!
free!

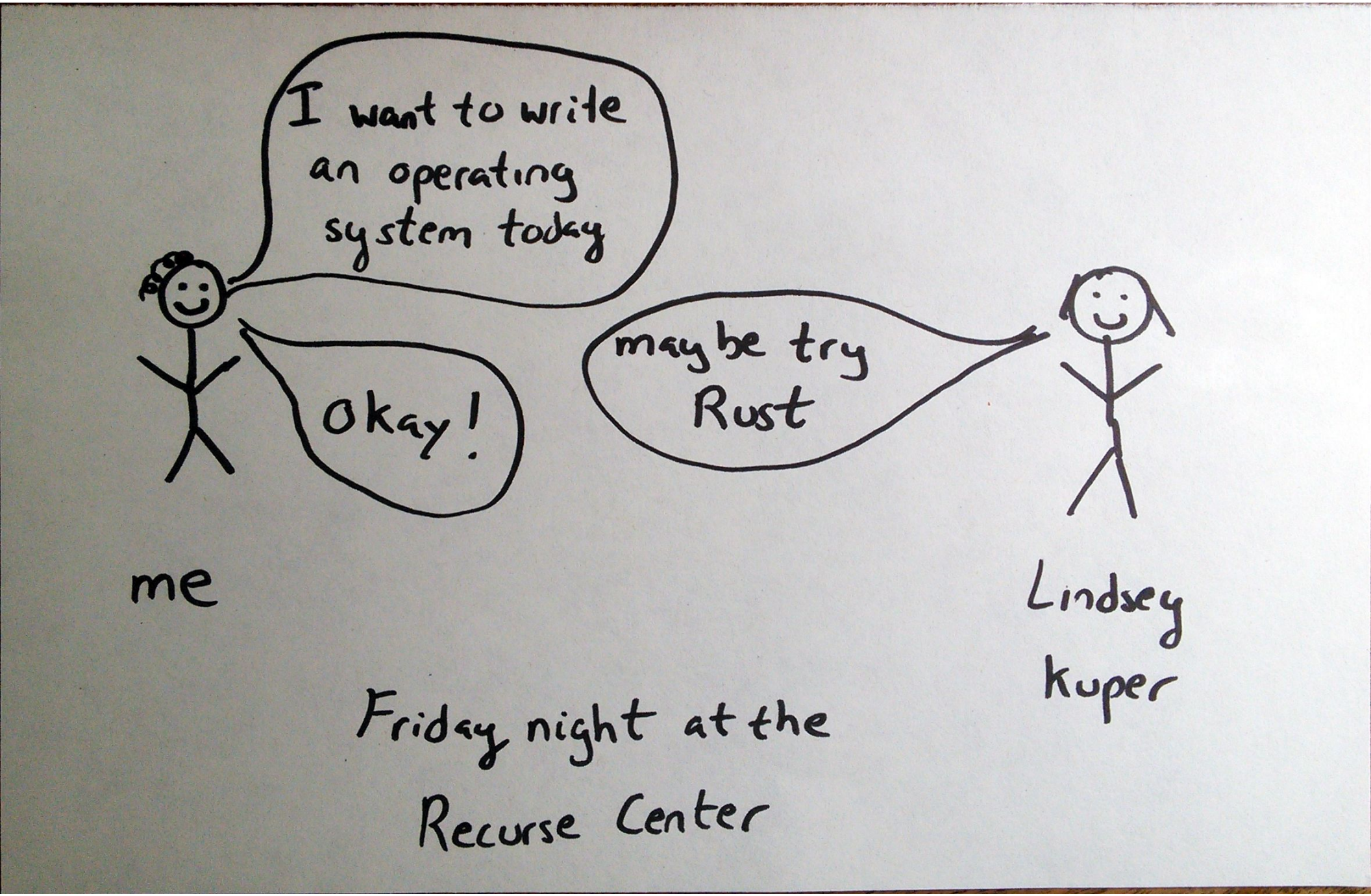


the best
programming
community
in the world

recurse.com


frontend
backend
machine learning
using a database
CSS
what's hard?
+
scary
web dev
Python
Java
any new language


OPERATING
SYSTEMS
!!
yay
amaze
WOW


I want to write an operating system today
Okay!
me
maybe try Rust
Lindsey Kuper
Friday night at the Recurse Center

Rust lets you write your own

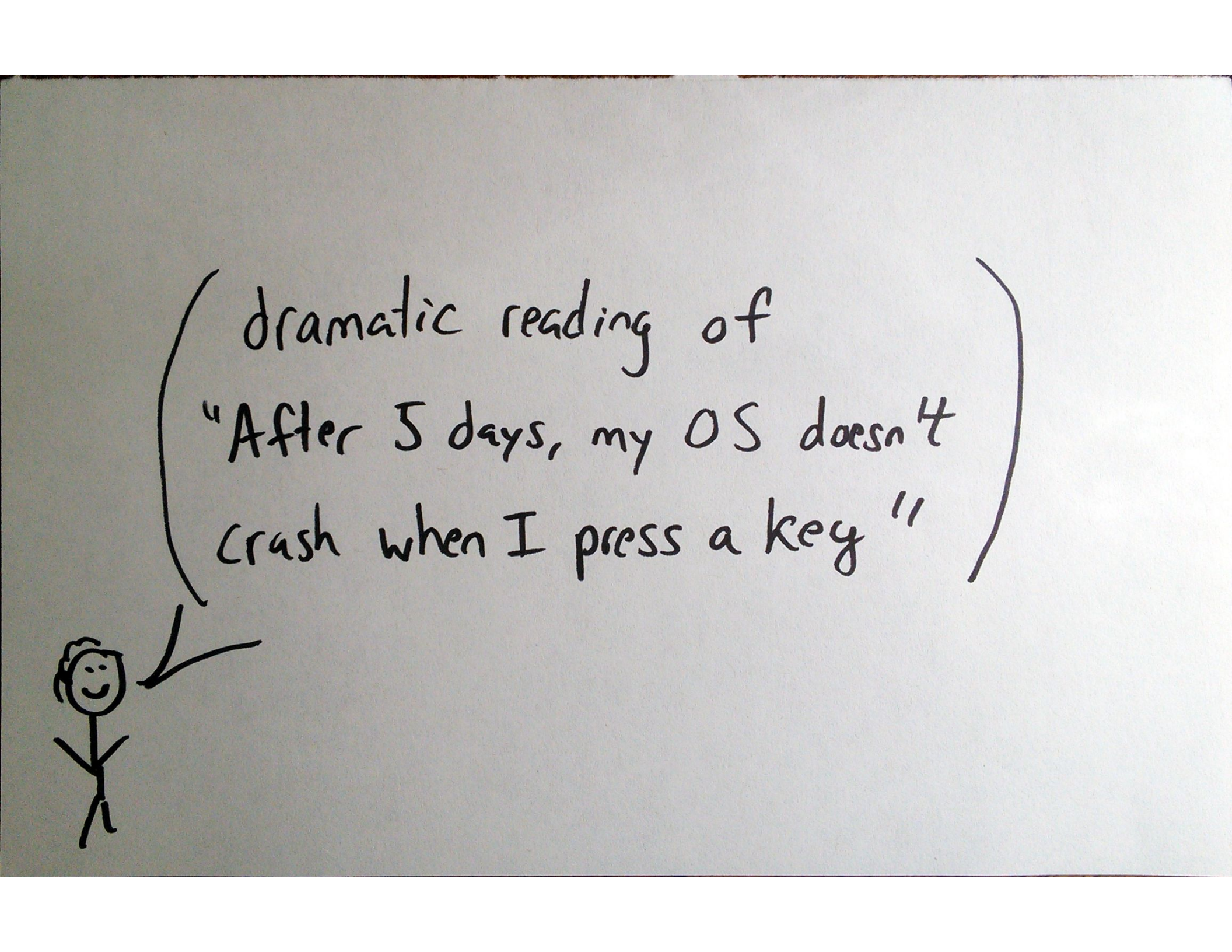
- malloc etc
- threads etc
- I/O

"~~freestanding mode~~"
libcore

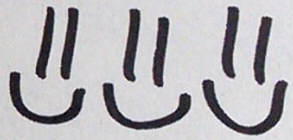
things you do not have when you write a kernel

- a hard drive
- files
- other processes
- a working keyboard
- memory protection
- any I/O
- libraries to use
- ANYTHING !!

"I'm just going to write a keyboard driver"


(dramatic reading of
"After 5 days, my OS doesn't
crash when I press a key")

remember how we don't

have malloc? 

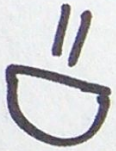
```
fn malloc(size: i32) {
    return 42;
}
```

now we have our weird malloc

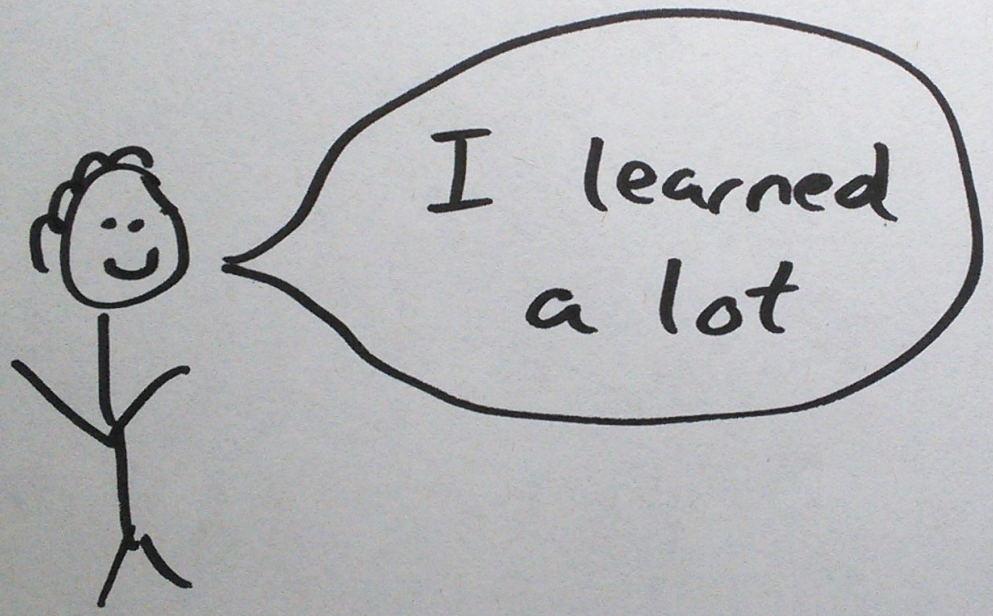
```
let x = Box::new(5)
let y = Box::new(88)
```

x=5

now x = 88



after 3 weeks




I'm trying to write a kernel and I have no idea what I'm doing also how do Rust strings work. halp.
we can help!
#rust

basically writing
a toy kernel is
REALLY FUN

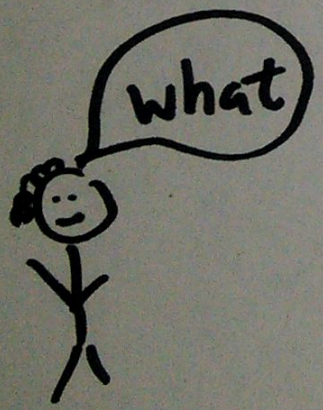
I learned SO MUCH.

In fact...

from: _@apple.com
to: julia@jvns.ca
subject: Opportunity with Apple

Hi,

I manage the Kernel Performance team at Apple...



WHAT'S HARD?
DATABASES!

and now I'm a

better programmer

← programmer crown