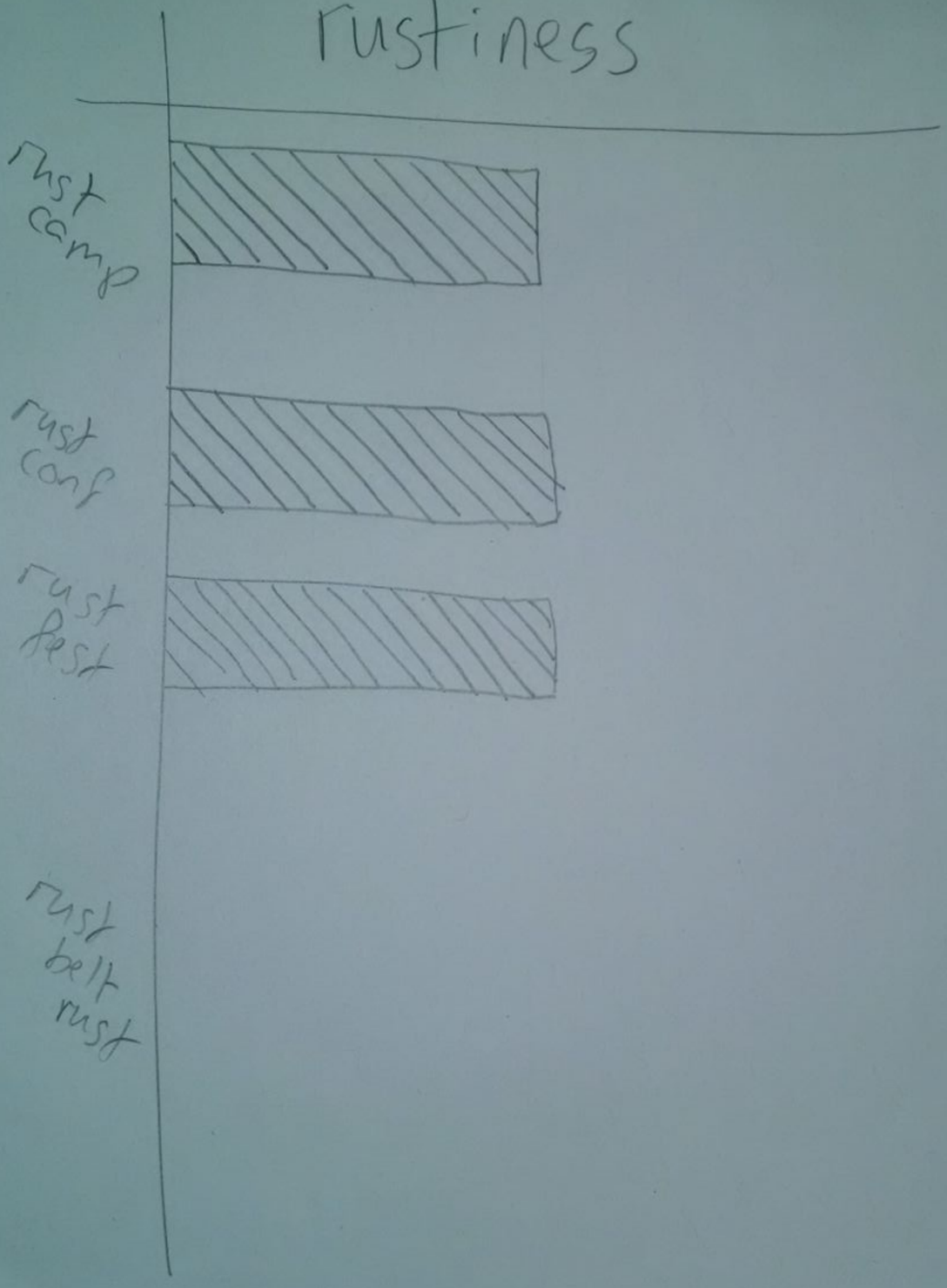
unix systems programming in rust

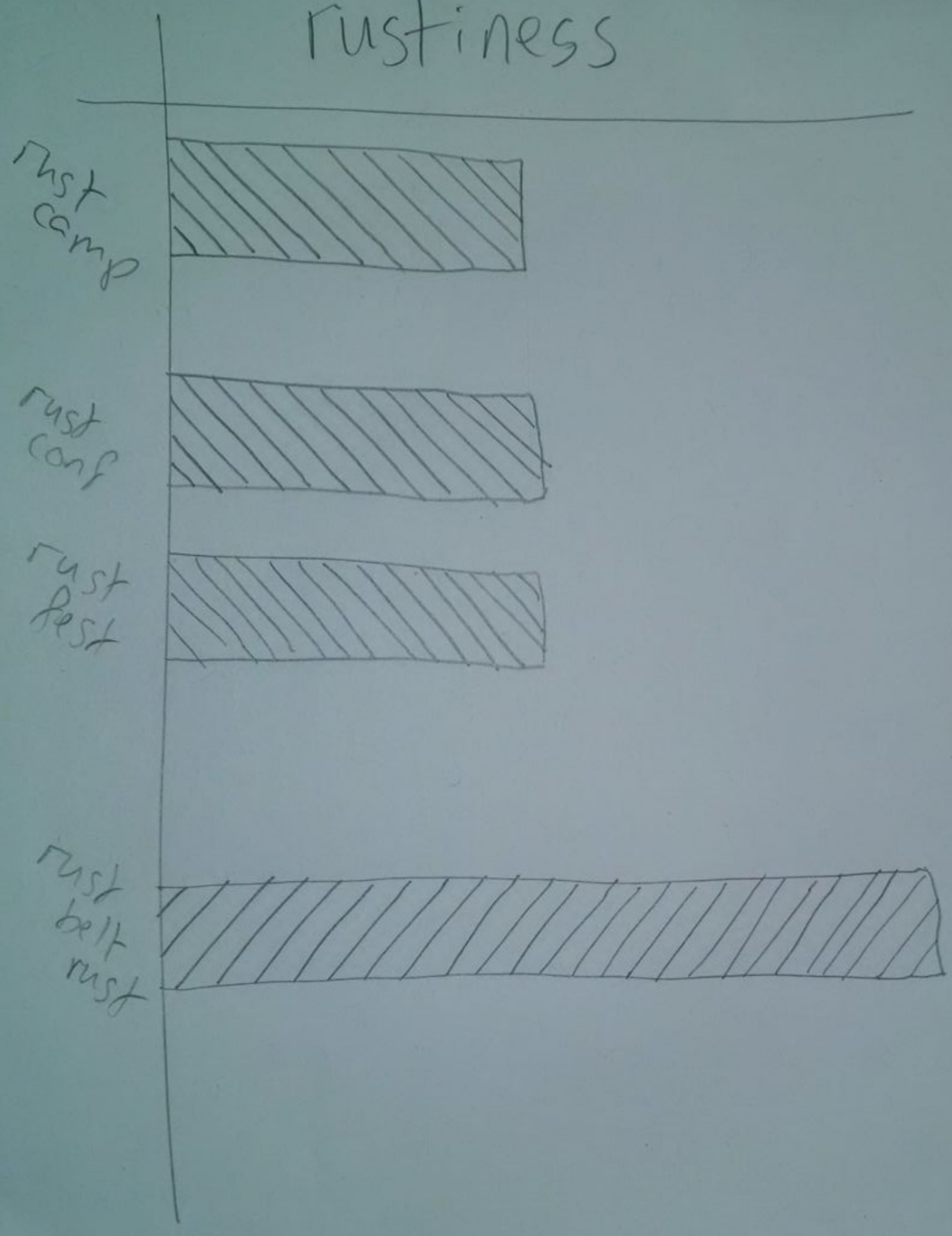
unix systems programming in rust is awesome

rust is awesome

rust belt rust

rust belt rust


rustiness
rust camp
rust conf
rust fest
rust belt rust


rustiness
rust camp
rust conf
rust fest
rust belt rust

rustiest rust conference

rustiest rust conference ever

systems programming in rust is awesome

systems
programming
is awesome

what do I mean by unix systems programming

more time looking at
man pages
than at
stack overflow

system calls

kernel's api

Dear Kernel,
please open gaels.gif for me.

Thanks,
<3 PID 14283


gi k·ne

gi k·ne

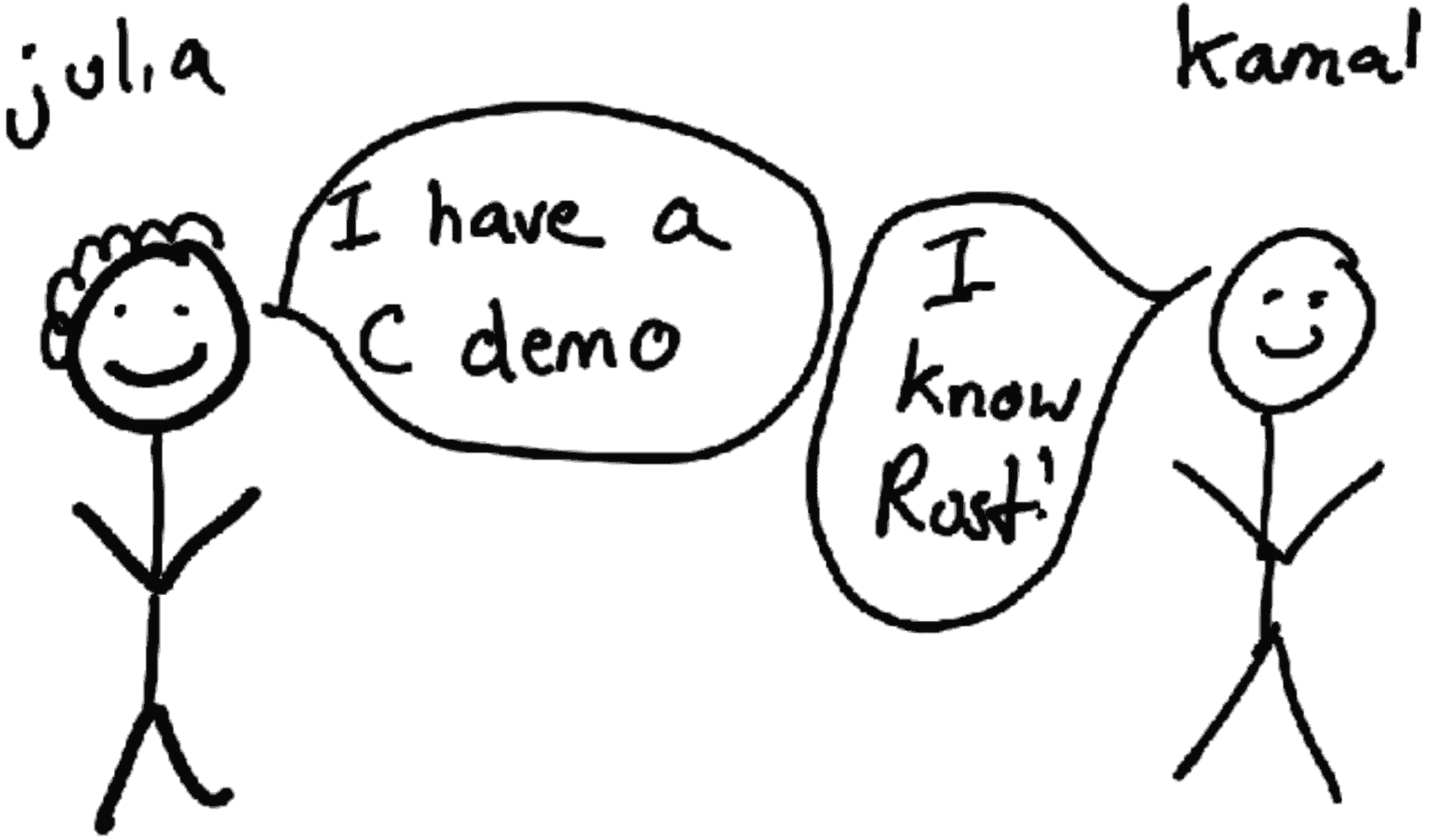
some system calls

- file operations: open, read, write
- processes: fork, exec, kill
- networking: socket, connect, listen, accept

examples of rust systems things I've worked on

containy-
thing

ruby-stacktrace


julia
kamal
I have a C demo
I know Rust!

silly things are ok too!

breaking my computer with pipes

why rust is great for systems programming

rust lets
you be
fearless

rust is convenient

rust allows transparent access to the underlying os

rust lets
you be
fearless

traditionally
done in c

c is scary if you don't know it well

c is scary if you don't know it well
(like me)

error checking
memory
management

allocate and free memory

no dangling pointers

rust is convenient

working with strings isn't painful

hash tables

hash tables!!!

crates.io

rust allows transparent access to the underlying os

std::os::unix

but we also get transparent access to c and libc

lets us drop down below libstd

os-
specific

containy-thing: unshare, mount

ruby-stacktrace: process_vm_readv


nix

rustifies libc

rustifies?
libc

oxidizes?
libc

makes libc nicer to use

Result instead of special return values and errno

we read the man pages so you don't have to

we read the man pages so you don't have to

(but you probably should!)

rust types where they make sense

tuples instead of int[2]

```
// C
int pipe_fds[2];
if (pipe(pipe_fds)) {
    return -1;
}
```

```
// Rust
let (rd, wr) = try!(pipe());
```

```
// Rust (soon)
let (rd, wr) = pipe()?;
```

&[u8] instead of (void*,size_t) pairs


<aside>

an ode to slices

standard way to pass pieces of buffers

standard means it's ecosystem-wide

contrast to c++

(start, length)
or (start, end)
pairs

but no standard

(c++17 will hopefully finally get string_view)

(but that's 2017)

(and you won't protected from lifetime mishaps)

rust had
slices from
the beginning

if all ownership
got us was safe
&str and &[u8] it
would be worth it

</aside>

nix is great

check it out!
crates.io/nix

how you can get started

write a shell!
http://j.mp/
shell-workshop

try some systems programming in rust with nix

you can
learn a lot by
being silly

how you can help

contribute to nix :-)

do cool things...

... and tell people about it!

write blog posts

give talks

because unix systems programming in rust really is awesome

thank you