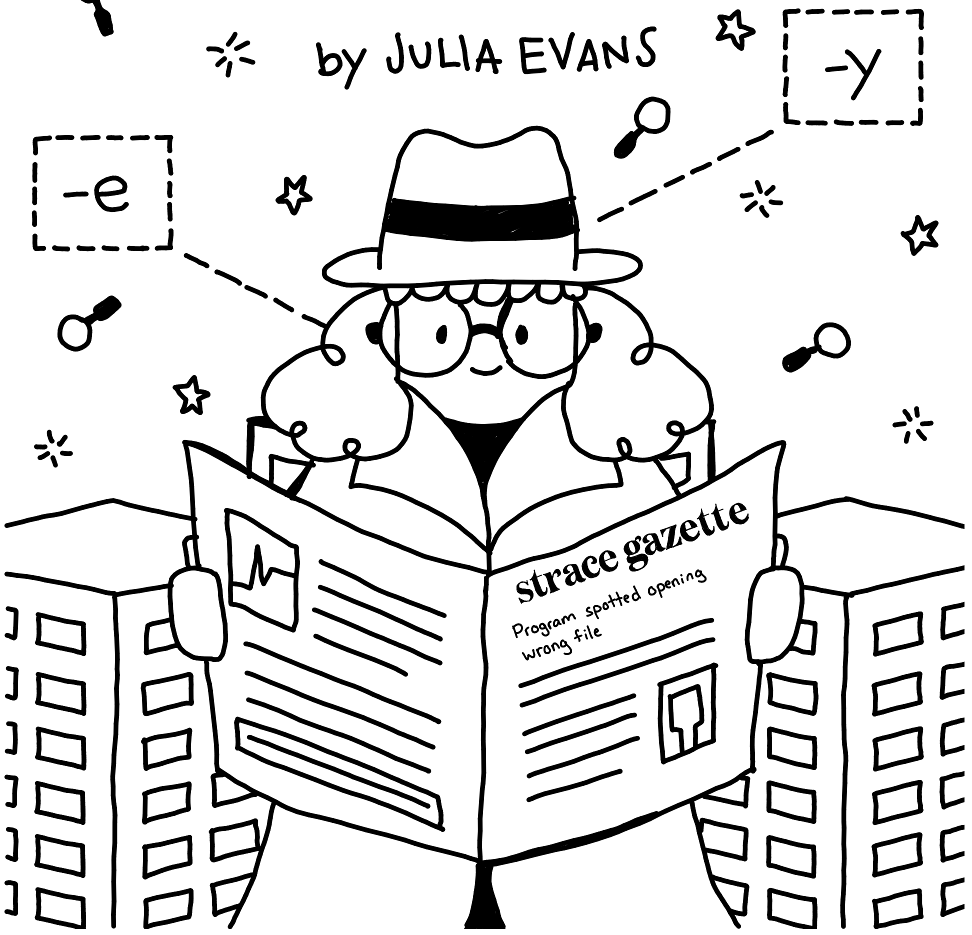
like this?
you can print more!
for free!
http://jvns.ca/zines

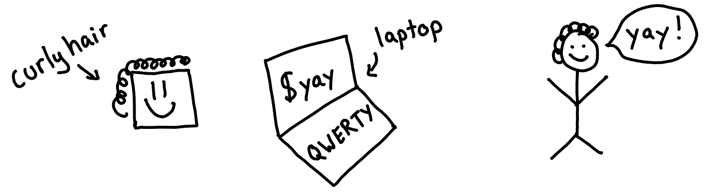
SPYING ON YOUR PROGRAMS WITH STRACE

by JULIA EVANS



Who makes this?

Hi! I'm Julia! I look kind of like this:



I found out last year that understanding your operating system's internals a little more makes you

and it was SO FUN and I wanted to tell EVERYONE. So I'm telling you! ⩌⩌⩌

I write more like this at

blog: jvns.ca
twitter: @b0rk
email: julia@jvns.ca

Resources + FAQ

I've written like 7 posts about strace because I have an unhealthy obsession. They're at

jvns.ca/categories/strace

(In)frequently asked questions:

Q: Is there strace on OS X?
A: No, but try dtruss/dtrace!

Q: Can I strace strace?
A: Yup! If you do, you'll find out that strace uses the ptrace system call to do its magic.

Q: Should I strace my production database?
A: NONONONO. It will slow down your database a LOT.

Q: Is there a way to trace system calls that won't slow down my programs?
A: Sometimes you can use `perf trace` on newer Linux versions

That's it! Now you're a WIZARD

More seriously, there's obviously a TON more to learn about operating systems and many further levels of wizardry. But I find just strace by itself to be an incredibly useful tool.

And so fun! On on a 12-hour train ride from New York to Montreal, I had no book and no internet so I just started stracing programs on my computer and I could totally see how the 'killall' program works without reading the source code or ANYTHING.

and it helps me debug all the time ♡

★ happy stracing ★

♡ a tiny manifesto ♡

operating systems are

AWESOME

the strace zine thinks:

- your computer is yours
- your OS is yours
- open licenses mean you can READ AND CHANGE THE CODE!!
- Linux is REALLY COOL

→→→→→→→→→→→→→→→→→→→→yaaaaay →→→→→→→

LET'S GO LEARN

→→→→→→→→→→→→→→→→it's really fun →→→→→→

what is this strace thing????

strace ← pronounced ess-trace

(on OSX you can use dtrace/dtruss)

strace is a program on Linux that lets you ~~inspect~~ spy on what a program is doing without

- a debugger
- or the source code
- or even knowing the programming language at all (?!!?! how can it be!)

Basically strace makes you a

WIZARD

To understand how this works, let's talk a little about operating systems

-s is for strings!!

Sometimes I'm looking at the output of a recvfrom and it's like

recvfrom(6, "And then the monster...")

and OH NO THE SUSPENSE

`strace -s 800` will show you the first 800 characters of each string. I use it all the time ★

-o is for output!

Let's get real. No matter what, strace prints too much damn output. Use

```
strace -o too_much_stuff.txt
```

and sort through it later.

-y

Have no idea which file the file descriptor "3" refers to? `-y` is a flag in newer versions of strace and it'll show you filenames instead of just numbers!

Putting it all together:

Want to spy on a ssh session?

```
strace -f -o ssh.txt ssh juliabox.com
```

See what files a Dropbox sync process is opening? (with PID: 230)

```
strace -f -p 230 -e open
```

strace command line flags I ♡

-e

overwhelmed by all the system calls you don't understand? Try

```
strace -e open
```

and it'll just show you the opens. much simpler ♡

-f

f is for follow

Does your program start subprocesses? lots do!

Use -f to see what those are doing too. Or just always use -f! That's what I do.

-p

p is for PID

"OH NO I STARTED THE PROGRAM 6 HOURS AGO AND NOW I WANT TO STRACE IT"

Do not worry! Just find your process's PID (like 747) and

```
strace -p 747
```

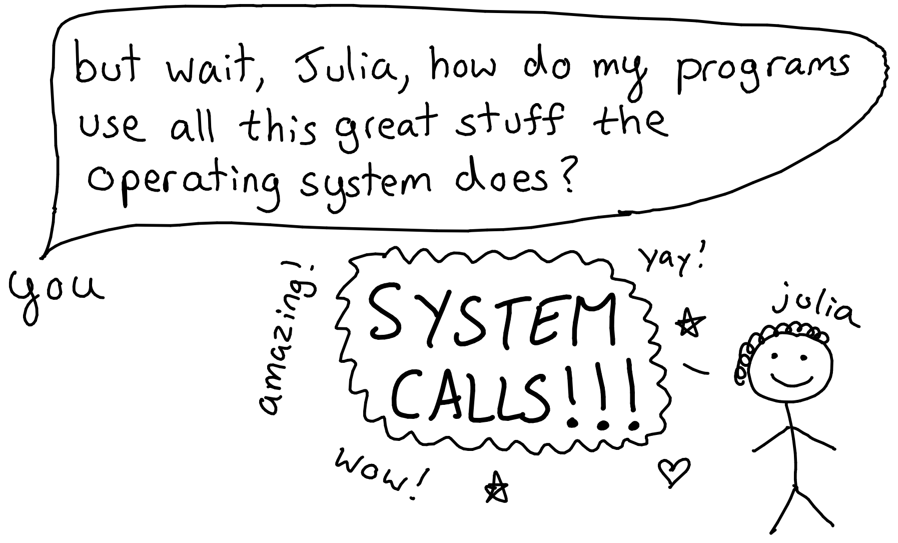
tip: if the process runs as root you'll need to be root too because SECURITY

Why you should ♡ your ★ operating system ★

Some things it does for you:

- understand how your hard drive works and how the filesystem on it organizes the bytes into files so you can just read your damn file ☺
- run code every time you press a key so that you can type
- implement networking protocols like TCP/IP so that you can get ~~webpages~~ pictures of cats from the internet
- keep track of all the memory every process is using!
- basically know everything about how all your hardware works so you can just write programs! ♡

so great


but wait, Julia, how do my programs use all this great stuff the operating system does?
you
SYSTEM CALLS!!!
amazing!
yay!
wow!
julia

System calls are the ~~API~~ interface for your operating system

- want to open a file? use `open` and then `read` and `write` to it

- sending data over a network? Use `connect` to open a connection and `send` and `recv` pictures of cats.

Every program on your computer is using system calls all the time to manage memory, write files, do networking, and lots more.

connect

Sometimes a program is sending network requests to another machine and I want to know WHICH MACHINE.

```
strace -e connect
```

shows me every IP address a program connects to.

What's fun? Spying on network activity is fun. If you have a HTTP service and you're debugging and totally at your wits' end, maybe it's time to look at what's REALLY EXACTLY being sent over the network...

these are your pals ♡

execve

On my first day of work, a Ruby script that ran some ssh commands wasn't working. Oh no!

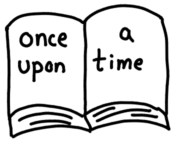
But who wants to read code to find out why? ugh.

```
strace -f -e execve ./script.rb
```

told us what the problem ssh command was, and we fixed it!

my fav♡rite system calls

open



Have you ever not been sure what configuration files a program is using? THAT NEVER NEEDS TO HAPPEN TO YOU AGAIN ☺☺☺. Skip the docs and head straight for:

```
strace -f -e open mplayer Rick_Astley.mp3
```

write

Programs write logs.

If you're sure your program is writing Very Important Information but don't know what or where, `strace -e write` may be for you.

`read` is pretty great too.

a first cup of strace

You might think with all this talk of operating systems and system calls that using strace is hard.

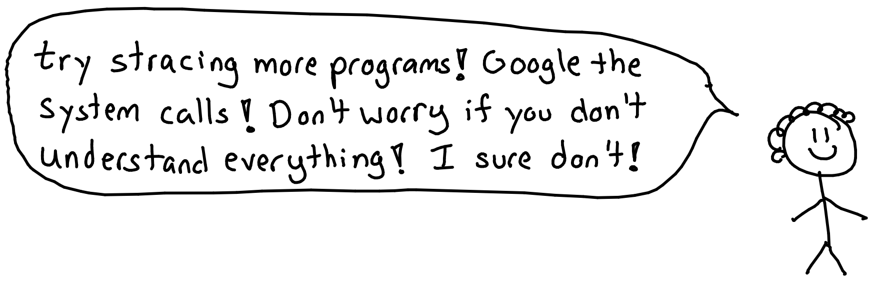
Getting started is easy! If you have a Linux machine I want you to try it RIGHT NOW.

Run:

```
strace ls
```

Wizard time!

There's a LOT of output and it's pretty confusing at first. I've annotated some for you on the next page ☺


try stracing more programs! Google the system calls! Don't worry if you don't understand everything! I sure don't!

annotated strace

When you run strace, you'll see thousands of lines of output like this:

```
$ strace ls /home/bork/blah
execve("/bin/ls", ["ls", "/home/bork/blah"], [/* 48 vars */]) = 0
brk(0)                                  = 0x172c000
stat("/usr/local/lib", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=180820, ...}) = 0
mmap(NULL, 180820, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe04e3f7000
close(3)                                = 0
open("/proc/filesystems", O_RDONLY)     = 3 fstat(3, {st_mode=S_IFREG|0444, st_size
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe04e423000
read(3, "nodev\tsysfs\nnodev\trootfs\nnodev\tr"..., 1024) = 334
read(3, "", 1024)                       = 0
close(3)                                = 0
stat("/home/bork/blah", {st_mode=S_IFDIR|0775, st_size=4096, ...}) = 0
openat(AT_FDCWD, "/home/bork/blah", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
getdents(3, /* 3 entries */, 32768)     = 80
getdents(3, /* 0 entries */, 32768)     = 0
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 4), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe04e423000
write(1, "awesome_file\n", 13)          = 13
close(1)                                = 0
munmap(0x7fe04e423000, 4096)            = 0
close(2)                                = 0
exit_group(0)                           = ?
```

Studies show this is not self-explanatory
(me asking my friends if it makes sense and NOPE NOPE)

★ let's learn how to interpret strace output ★

$\underbrace{\text{11999}}_{①}\ \underbrace{\text{execve}}_{②}(\underbrace{\text{"/usr/bin/ssh", ["ssh", "jvns.ca"]}}_{③}) = \underbrace{0}_{④}$

① The process ID (included when you run strace -f)

② The name of the system call (execve starts programs!!)

③ The system call's arguments, in this case a program to start and the arguments to start it with

④ The return value.

still the name of the syscall ↓ — file to open ↓ — open with read/write permissions ↓

open("awesome.txt", O_RDWR) = 3 ← file descriptor

The 3 here is a file descriptor number. Internally, Linux tracks open files with numbers! You can see all the file descriptors for process ID 42 and what they point to by doing

`ls -l /proc/42/fd`   'fd' is for file descriptor!

file descriptor ↓ — what got read ↓ — number of bytes read ↙

read(3, "wow! yay!") = 9

If you don't understand something in your strace output:

- it's normal! There are lots of syscalls.
- try reading the man page for the system call!

  `man 2 open`

- remember that just understanding read + write + open + execve can take you a long way ♥