how to spy on your programs with strace

in which we learn a bout...

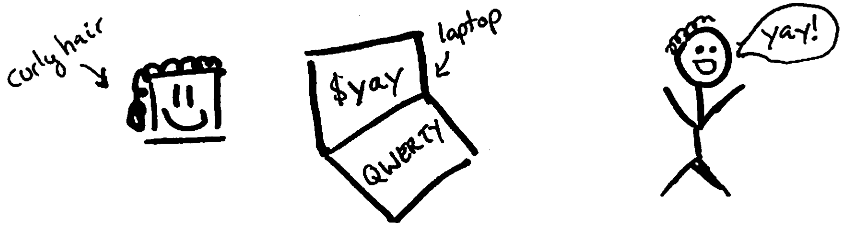
- ★ how one standard Linux utility can make you a 'WIZARD' (it's strace ☺)
- ★ why you should ♥ your ♡operating♡ ♡system♡
- ★ that system calls are THE BEST (and what my favourites are !!)

$5.00 or trades ♥

Julia Evans, strace wizard wow fun yay industries 2015

who makes this?

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



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

WAY BETTER PROGRAMMER

AWESOME WOW YAY

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

http://jvns.ca/categories/strace

(In)frequently asked questions:

Q: Is there strace on OSX?
A: No, but you can use dtrace/dtruss and it's actually much more powerful!

Q: Can I strace strace?
A: Yup! It uses the ptrace system call.

Q: Can I strace PID 1 (init)?
A: APPARENTLY YES! (use extreme caution ☺)

Q: Should I strace my production database?
A: NONONONO. It will run MUCH more slowly never do this.

That's it! Now you're a WIZARD

more seriously obviously there's 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! Once on a 12-hour train ride fron 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 killall worked without reading the source code or ANYTHING.

also 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**
- **just because some Linux kernel devs (cough Linus cough) act like jerks doesn't mean we can't still learn AWESOME STUFF** ♡

→ → → → → → → yaaaaaay → → → →

LET'S GO LEARN it's really fun → → → →

What is this strace thing????

on OS X you can use dtrace

♡♡♡ `strace` is a program on <u>Linux</u> that lets you ~~inspect~~ spy on what a program is doing without

- a debugger
- or the source
- 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 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

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.

Putting it all together:

Let's say you wanted to spy on a ssh session!

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

Or see what files a Dropbox sync process is opening (made up 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? lols
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

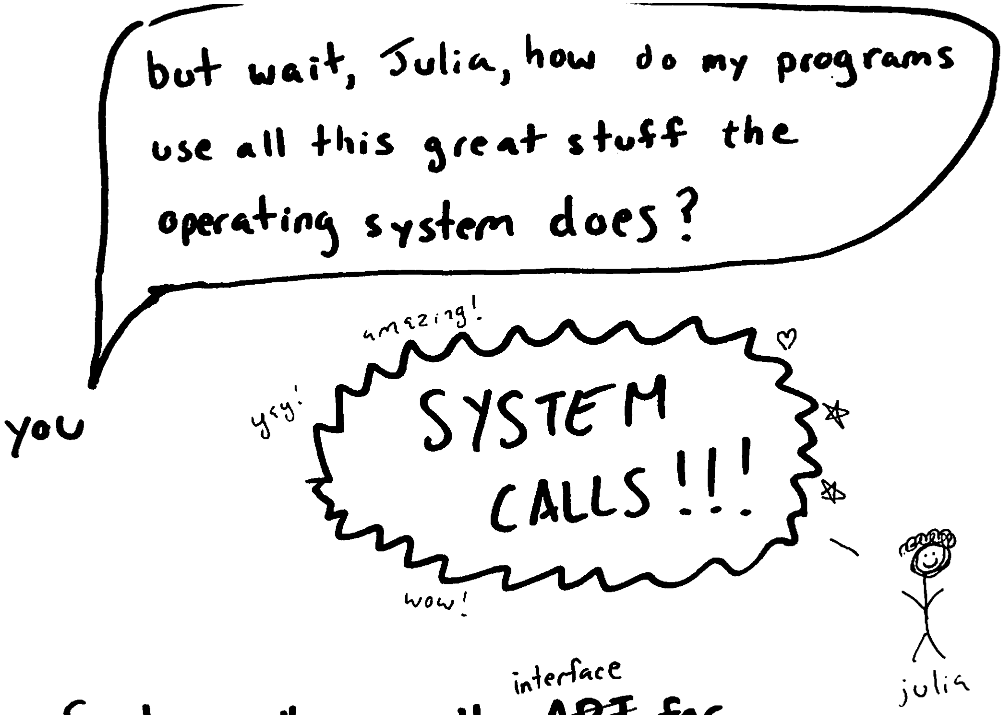
```
strace -p 747
```

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 every thing about how all your hardware works so you can just write programs! ♥


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

- Send ~~to~~ data over a network? Use socket to open a connection and sendto and recvfrom pictures of cats

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

010100 1111 0 010

sendto + recvfrom

0100010100000

What's fun? Spying on network activity is fun. If you have a HTTP service ~~or~~ 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 ♡

★ note: network activity can show up in read and write ~~examples~~ syscalls too. We saw that in the SSH example!

execve

program executions!

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
```

psst: I'm going to explain -e and -f in a couple of pages ☺

write

Programs write logs.

```
write(f, "OH NOEZ");
```

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.

a first cup of strace

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

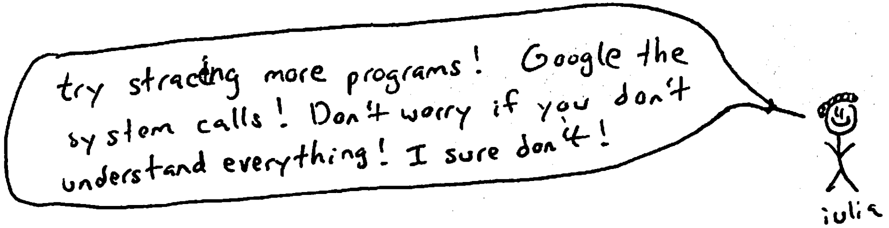
It's easy! If you have a Linux machine I want you to try it RIGHT NOW

```
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 ☺

because I ♡ examples


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

annotated strace

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

```
execve("/bin/ls", ["ls"], [/* 50 vars */]) = 0
brk(0)                                  = 0x131f000
open("/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\340\r\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=14768, ...}) = 0
close(3)                                = 0
mprotect(0x7fa58e19f000, 2093056, PROT_NONE) = 0
set_tid_address(0x7fa58f39ea90)         = 7350
set_robust_list(0x7fa58f39eaa0, 0x18)   = 0
futex(0x7fff6335ed2c, FUTEX_WAIT_BITSET_PRIVATE|FUTEX_CLOCK_REALTIME, 1, NULL, 7fa58f39e7c0) = -1
rt_sigaction(SIGRTMIN, {0x7fa58e3a6750, [], SA_RESTORER|SA_SIGINFO, 0x7fa58e3afcb0}, NULL, 8) =
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
getrlimit(RLIMIT_STACK, {rlim_cur=8192*1024, rlim_max=RLIM_INFINITY}) = 0
statfs("/selinux", {f_type="EXT2_SUPER_MAGIC", f_bsize=4096, f_blocks=14385663, f_bfree=5356302,
f_frsize=4096}) = 0
brk(0)                                  = 0x131f000
brk(0x1340000)                          = 0x1340000
open("/proc/filesystems", O_RDONLY)     = 3
read(3, "nodev\tsysfs\nnodev\trootfs\nnodev\tr"..., 1024) = 345
mmap(NULL, 7257616, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fa58daaf000
close(3)                                = 0
ioctl(1, SNDCTL_TMR_TIMEBASE or TCGETS, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, TIOCGWINSZ, {ws_row=40, ws_col=144, ws_xpixel=0, ws_ypixel=0}) = 0
openat(AT_FDCWD, ".", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
getdents(3, /* 19 entries */, 32768)    = 608
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 3), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa58f3cf000
write(1, "build\t\t dist  LICENSE\tperf.data "..., 99) = 99
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
```

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

★ let's learn how to interpret strace output ★

$\underbrace{\text{11499}}_{①}\ \underbrace{\text{execve}}_{②}\underbrace{\text{("/usr/bin/ssh", ["ssh", "jvns.ca"]}}_{③}\text{...}$

① The process ID

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 (invisible, at the end) The return value.

Let's explain just a couple more things!

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

open("/awesome.txt", O_RDWR) = 3

The 3 here is a file descriptor number. ~~which~~

Internally Linux tracks 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 get it

file descriptor ↓ · what got read ↓ · # bytes read ↓

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

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

- me too! It's normal!
- try reading the man page for the system call!
- remember that just understanding read/write/open/execve can take you a long way ♥