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
Happy steering

Linux is really cool.
- Read and change the code!
- Open licenses mean you can
- Your OS is yours
- Your computer is yours

The strange things:
- awesome
- Operating systems are
  - A tiny manifesto

That's it! Now you're a wizard.

And I helped me debug all the time.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.

And so fun! On a 12-hour train ride from New York to Montreal, I had no book and no internet.

I just started staring at the program works so I could totally see how the Killall program works without reading the source code or anything.
what is this strace thing????

Strace is a program on Linux that lets you inspect 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.

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 ★

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.

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 julia.box.com

See what files a Dropbox sync process is opening?

(strace -f -p 230 -e open)
Programs work because you can just write to memory. Know everything about how all your programs work and basically everything about how the operating system works.

- Keep track of all the memory every process is using.
- From the internet or from the documentation, get a list of all the files or network protocols like TCP/IP so that you can type 'sudo fsck -t' every time you press a key so the kernel reads your damn files.

Some things it does for you:

- It keeps track of all the memory every process is using.
- It provides a list of files and network protocols like TCP/IP so that you can type 'sudo fsck -t' every time you press a key so the kernel reads your damn files.

Why you should do your:

- `strace -p 747`
- `pkill -f` and `strace IT "OH NO I STARTED THE PROGRAM 6 HOURS AGO AND NOW I WANT TO " OR just always use `pkill -f` to see what those are doing too.
- `use -f` to see what those are doing too.

Does your program start `$pkill -f`? Or does it just show you the output? Try `strace e: open` to understand what it calls overwhem by all the system calls.

Flags I:

- `-p`
but wait, Julia, how do my programs use all this great stuff the operating system does?

you: amazing!

yay!

Julia: System calls!!!

Wow! 😊

System calls are the API 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.

```bash
$ 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.

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

told us what the problem ssh command was, and we fixed it!
A first cup of Strace

```
Run: Strace is Wizard
```

For you on the next page.

```
System calls? Don't worry if you don't try tracing more programs? Google the reads. Is pretty great too.
```

Terry is a lot of output and it's pretty confusing at first. I've annotated some programs write logs.
annotated strace

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

```c
$ strace ls /home/bork/blah
execline("/bin/ls", ["ls", "/home/bork/blah"], [/* 48 vars */]) = 0
brk(0) = 0x17c000
stat(1112, (st_mode=S_IFDIR|0755, st_size=180020, ...)) = 0
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, (st_mode=S_IFREG|14044, st_size=180020, ...)) = 0
map(NULL, 180020, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7e0e8e3f7000
close(3) = 0
open("/proc/filesystems", O_RDONLY) = 3
fstat(3, (st_mode=S_IFREG|14044, st_size=180020, ...)) = 0
map(NULL, 180020, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7e0e8e423000
read(3, "nodev/千年/nnodev/trootfs/nnodev/tr", ..., 1024) = 334
read(3, "", 1024) = 0
close(3) = 0
stat("/home/bork/blah", (st_mode=S_IFDIR|1777, st_size=180020, ...)) = 0
ownat(AT_FDCWD, "/home/bork/blah", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
getdents(3, /* 3 entries */ 32768) = 32
getdents(3, /* 6 entries */ 32768) = 0
close(3) = 0
fstat(1, (st_mode=S_IFCHR|1629, st_rdev=mkdev(136, 4, ...)) = 0
map(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7e0e8e423000
write(1, "awesome_file\n", 13) = 13
close(1) = 0
munmap(0x7e0e8e423000, 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 ★

1. **execline**("/usr/bin/ssh", ["ssh", "jans-as"]
2. The process ID (included when you run strace -f)
3. The name of the system call (execline starts programs)
4. The system call's arguments, in this case a program to start and the arguments to start it with
5. The return value.

Still the name of the syscall ➔ file descriptor ➔ file to open ➔ read/write permissions ➔ `open("awesome.txt", O_RDONLY)=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
```

The `fd` is for file descriptor!

```
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 + execline can take you a long wayrence that just understanding read + write + open + execline can take you a long way