Motivation
Broken date
Broken Troff
Just use acid, right?
How it works
Comparing ratrace and strace
Performance
Summary

System call tracing

Ron Minnich

Fifth IWP9
With thanks to Russ Cox, Jim McKie, and Noah Evans

Outline

1 Motivation

2 Broken date

3 Broken Troff

4 Just use acid, right?

5 How it works

6 Comparing ratrace and strace

7 Performance

8 Summary

Something doesn't work. Why?

- It's hard to find out
- Especially if there is an rfork
- An example .. what if on-stack variables break?

broken date

- Post-july 2010 build on 9vx from updated source had problems
- Date would just do nothing
- get impatient, hit return, it worked
- Acme had strange hangs
- Nothing worked

broken date

```
24577 date Pread 0x19f6 0 0ffffee0/"." 8 0 =
1 "" 0x11cef69ae0b06c68 0x11cef69c0a58d900
24577 date Close 0x1a30 0 = 0 ""
0x11cef69c0b601f70 0x11cef69c0b607948
24577 date Open 0x1a89
0000702c/"/dev/bintime" 00000020 = 0 ""
0x11cef69c0c2119c8 0x11c ef69c0c5911e8
etc.
```

- Read from fd 0? What?

What? : off to code

broken troff

```
term% troff -ms troff.ms | page converting
from troff to postscript...
.
.
reading through postscript... postnote 1828:
sys: write on closed pipe pc=0x0001f8fc term%
```

- I was not even sure where to start with that one
- It's a shell pipeline
- And somehow the failure on exec got lost

Er, what?

- Hard to follow this kind of thing.
- But:
- ratrace -c /bin/rc -c “troff -ms troff.ms | page”
- and wait a bit ...

```
1938 page Exec 0x2eac 0x279e6/"/bin/gs"
0x27973/"gs" 0x27976/"-dNOPAUSE"
0x223f1/"-dSAFER"
0x27980/"-sDEVICE=plan9"
0x2798f/"-sOutputFile=/fd/3"
0x279a2/"-dQUIET" 0x279aa/"-r100"
0xdfffcda8/"-dTextAlphaBits=4"
0xdfffcd88/"-dGraphicsAlphaBits=4"
0x279da/"-" = ffffffff
```

- So the problem was not the one you might have thought: it was just that gs was gone
- But the error from exec got ignored (only observed in the child)
- And the parent got the EPIPE

Just use acid, right?

- It doesn't always work (Blue Gene, 9vx, sometimes ARM)
- But even if you could get it to work
- Requires a number of local files (painful for ram disk setup)
- But even if you had local files
- It's a pain with fork
- And even if it was not a pain with fork
- Well, I just don't find it as convenient as ratrace

How it works

- Needed a way to tell a program to stop on system call entry
- Oh wait, it's already there!
- Need to pretty-print system call args etc.
- 9vx led the way
- 9vx showed that one could dump system calls with a simple “boot time” option
- So the key was generalize it, make it prettier, make it a device

Generalize it

- The 9vx printing was pretty raw and went to the console
- Needed to attach the string to a device
- So, add a new struct member to proc

New code

- New proc struct
- New code in syscall
- New code in devproc
- New code in proc
- let's go look

And the ratrace program itself

- A note on interface design
- We can not criticize original ptrace design
- The fact that we are using it forty years later, well, that we might be a little harsh about
- What does ptrace lead to in real life?
- strace /bin/date
- strace strace /bin/date

tracing strace

```
rminnich@ratnet:/$ strace /bin/date 2>/tmp/x
Wed Oct 6 15:55:21 PDT 2010
rminnich@ratnet:/$ wc /tmp/x
71 426 4727 /tmp/x
rminnich@ratnet:/$ strace 2>/tmp/xx strace
/bin/date
Wed Oct 6 15:55:41 PDT 2010
rminnich@ratnet:/$ wc /tmp/xx
1770 12163 99974 /tmp/xx
```

- /bin/date: 71 system calls
- strace /bin/date: 1770 system calls
- Factor of 24 blowup
- strace itself is 58KLOC
- Most of those calls look like this

What strace looks like

```
wait4(4294967295, [{WIFSTOPPED(s) &&
WSTOPSIG(s) == SIGTRAP}], __WALL,
NULL) = 13557
rt_sigprocmask(SIG_BLOCK, [HUP INT
QUIT PIPE TERM], NULL, 8) = 0
ptrace(PTRACE_PEEKUSER, 13557,
8*ORIG_RAX, [0x9]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*CS,
[0x33]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*RAX,
[0xffffffffffffffda]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*RDI,
[0]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*RSI,
[0x1000]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*RDX,
[0x3]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*R10,
[0x22]) = 0
ptrace(PTRACE_PEEKUSER, 13557, 8*R8,
```

what ratrace looks like

```
term% ratrace -c /bin/date
2054 8.ratrace Open 0x10f9
0x1dcc8/"/proc/2056/ctl"
2054 8.ratrace Pwrite 0x3d1e 3
11b8e/"startsyscall" 12 -0x1 = 0 ""
0x11da80e80b453ff4 0x11da80e80bfbb71e
```

and a Pread, and a Pwrite
(note: ratracing ratrace is broken in sources, help welcome :-)

- Approximately four system calls per syscall.

Timing

```
term% time ratrace -c /bin/date
0.00u 0.00s 0.15r ratrace -c /bin/date
term% time /bin/date
Thu Oct 7 13:37:31 PDT 2010 0.00u 0.00s
0.01r /bin/date
term%
```

- Dumb example

```
0.00u 0.22s 1.07r ratrace -c /bin/wc
/rc/bin/0a /rc/bin/0c ...
term% time wc /rc/bin/* > /dev/null
0.00u 0.10s 0.12r wc /rc/bin/0a /rc/bin/0c
/rc/bin/0l /rc/bin/9fat: ...
term%
```

- Looks like a factor of 10
- About the same overhead as for strace

Motivation
Broken date
Broken Troff
Just use acid, right?
How it works
Comparing ratrace and strace
Performance
Summary

System call tracing is easy and can shorten problem resolution

- Addition to Plan 9 was very straightforward
- Proper interface design leads to compact program design and compact operation
- Text interface is a good thing