



Writing SystemTap Scripts

Eugene Teo

Slides: <http://www.kernel.sg/talks/lca2008/>

What is SystemTap?

- Troubleshooting and analysis tool
- SystemTap scripting language translator
- Protected and simple interface to kprobes
- System-wide and process-centric views
- Extensible framework to write new tools
- Free and open source software (GPL)

Why use SystemTap?

- To extend functionality in traditional tools
- To find out system or process-related queries
 - Why does my disk keep writing something?
 - What is the current resource limits for mingetty?
 - How many sockets and pipes did dbus-launch use?
 - How can I pin down a lock contention problem?
- To write new tools and instrumentation frameworks
 - I like using tool X but it is only available in operating system Y
 - Fault injection framework

Why use SystemTap?

- Learn by looking at examples:
 - `kprobe_example.c` in `linux-2.6/Documentation/kprobes.txt`
 - SystemTap equivalent – `kprobe_example.sh` + `regs.stp`
- Compare the two different implementations
 - Complexity
 - Amount of code
 - Code safety
- Write less code yet achieve the same objective

How SystemTap works?

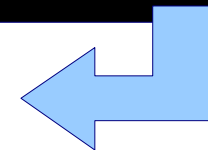
- SystemTap tool goes through 5 passes:
 - pass 1: parsing
 - pass 2: semantic analysis
 - pass 3: translation
 - pass 4: compilation
 - pass 5: run

Components of SystemTap script

- Strictly typed, declaration free, procedural, and inspired by awk
- Comments
... shell style, // ... C++ style, /* ... C style */
- global var1, var2 [= <value>]
- if (EXP) STMT1 [else STMT2]
- do STMT while (EXP)
- for (EXP1; EXP2; EXP3) STMT
- foreach (VAR in ARRAY) STMT
- <<<, @count, @avg, @hist_linear, @hist_log

```
Allocations by size in bytes
```

value	-----	count
2		0
4		0
8	@@@@@@@@@@@@@@@@	15
16		0
32	@@@@@@@@@@@@@@@@	15
64	@@@@@@@@@@@@@@@@	15
128		0
256		0



Components of SystemTap script

- Probes
 - probe kernel.function(PATTERN1) { [stmt ...] }
 - PATTERN1 can be “function@path/to/file.c:123”
 - probe module(PATTERN2).function(PATTERN1) { [stmt ...] }
 - PATTERN2 can be “ext3” or “scsi_mod”, etc.
 - probe netdev.transmit { [stmt ...] }
- Functions
 - function translate_mask (mask) { <stmt ...> }
 - function get_task_struct:long (pid:long) %{ <C_stmts ...> %}

Let's look at kprobe_example.sh

- The SystemTap script is called within the shell script
 - addr is gotten from /proc/kallsyms via the shell script
 - get_eip_info() and get_eflags_info() are from regs.stp tapset
- Recap: tapset = reuse = extensible framework, no Makefiles

Components of SystemTap Tapset

- Think of this as a library
- Reusable probes and functions
- Encapsulate knowledge of kernel subsystem
- Same syntax as a SystemTap script
- Also ends with .stp; located at /usr/share/systemtap/tapset
- Use `stap -I DIR` to specify path to additional tapset scripts
- ```
$ grep stap kprobe_example.sh
/usr/bin/env stap -p$1 -I ./ -e '
```

# Let's look at regs.stp

- Additional helper routines that can be reused by other scripts
- kread() macro – defined in runtime/loc2c-runtime.h
  - To dereference pointers that could potentially be invalid
  - Use it with CATCH\_DEREF\_FAULT()
- Recap: code safety, reuse and reduce code, reduce complexity

# Getting started

- If it is not already installed, run:
  - `yum install systemtap kernel-devel`
  - `yum -enablerepo=fedora-debuginfo install kernel-debuginfo`
- Or try out bleeding edge version:
  - read `src/README`
- System Administrators – Use existing SystemTap scripts
  - <http://sourceware.org/systemtap/wiki/WarStories>
  - `src/examples`, `src/examples/small_demos`
- Developers – Learn from existing scripts, write your own or reuse

Some interesting  
(and useful) scripts

# nettop.stp

- Defined netdev.transmit and netdev.receive probes
- Prints network activity every 5 secs
- simpres sends and receives packets sometimes. Why?

# iotop.stp

- Prints I/O activity

```
eteo@kerndev:~/lca
File Edit View Terminal Tabs Help
500 3440 1 gnome-terminal R 43657
500 19868 19840 vpngui R 16064
500 4002 3972 vpngui R 16064
500 3475 3446 vpngui R 16064
500 3217 3111 metacity R 8940
0 2389 1 rsyslogd W 7200
0 2393 1 rklogd W 6594
0 2393 1 rklogd R 4271
500 10494 19868 ifconfig R 2608

Thu Jan 24 23:52:18 2008 , Average: 51Kb/sec, Read: 242Kb, Write:

UID PID PPID CMD T BYTES
0 3047 3042 X R 146291
500 19868 19840 vpngui R 16064
500 4002 3972 vpngui R 16064
500 3475 3446 vpngui R 16064
500 3440 1 gnome-terminal R 12910
0 2389 1 rsyslogd W 7200
0 2393 1 rklogd W 6594
0 2393 1 rklogd R 4271
500 10500 19868 ifconfig R 2608
500 10501 19868 ifconfig R 2608
```

```
eteo@kerndev:~
File Edit View Terminal Tabs Help
post_handler: p->addr=0xc042c96b, eflags=0x106
pre_handler: p->addr=0xc042c96b, eip=c042c96c, eflags=0x206
[<c042c96b>] do_fork+0x0/0x17e
[<f8d7806a>] handler_pre+0x29/0x2f [kprobe_example]
[<c042c96b>] do_fork+0x0/0x17e
[<c042c96c>] do_fork+0x1/0x17e
[<c061fd40>] kprobe_exceptions_notify+0x164/0x386
[<c06209ab>] notifier_call_chain+0x2a/0x47
[<c0453b7d>] stopmachine+0x0/0x9a
[<c06209e6>] atomic_notifier_call_chain+0x17/0x1a
[<c0440c41>] notify_die+0x30/0x37
[<c061f310>] do_int3+0x2d/0x75
```

```
root@kerndev:~
File Edit View Terminal Tabs Help
e,snd_pcm_oss,snd_mixer_oss,snd_pcm,snd_timer,snd_hwdep
joydev 11649 0
sr_mod 17509 0
soundcore 9633 2 snd
i2c_core 21825 1 i2c_i801
cdrom 33889 1 sr_mod
dm_snapshot 17765 0
dm_zero 5825 0
dm_mirror 21569 0
dm_mod 46209 9 dm_multipath,dm_snapshot,dm_zero,dm_mirror
ata_piix 16709 2
ata_generic 8901 0
libata 100145 2 ata_piix,ata_generic
sd_mod 27329 7
scsi_mod 119757 5 usb_storage,sg,sr_mod,libata,sd_mod
ext3 110665 4
jbd 52457 1 ext3
mbcache 10177 1 ext3
uhci_hcd 23633 0
ohci_hcd 21445 0
ehci_hcd 31821 0
[root@kerndev ~]# lsmod | grep kprob
kprobe_example 5952 0
[root@kerndev ~]#
```

# pfiles

- Very nice tool, inspired by Solaris' pfiles tool
- Maps process PID to file descriptors
- Displays open socket information
- Lists opened files that are locked

# plimit

- Inspired by Solaris' plimit tool



# psig

- Inspired by Solaris' psig tool

# sig\_by\_proc.stp

- Prints signal counts by process name in descending order

# socktop

- Inspired by Dtrace's socktop
- Tracks reads and writes on sockets by process

# What's next

- Wiki: <http://www.sourceware.org/systemtap/wiki>
- Language reference: <http://sourceware.org/systemtap/langref/>
- Mailing list: [systemtap@sources.redhat.com](mailto:systemtap@sources.redhat.com)
- My blog: <http://www.kernel.sg/blog/category/systemtap/>
- This work is licensed under a [Creative Commons Attribution-Share Alike 3.0 Unported License](#).





## Writing SystemTap Scripts

Eugene Teo

Slides: <http://www.kernel.sg/talks/lca2008/>