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?

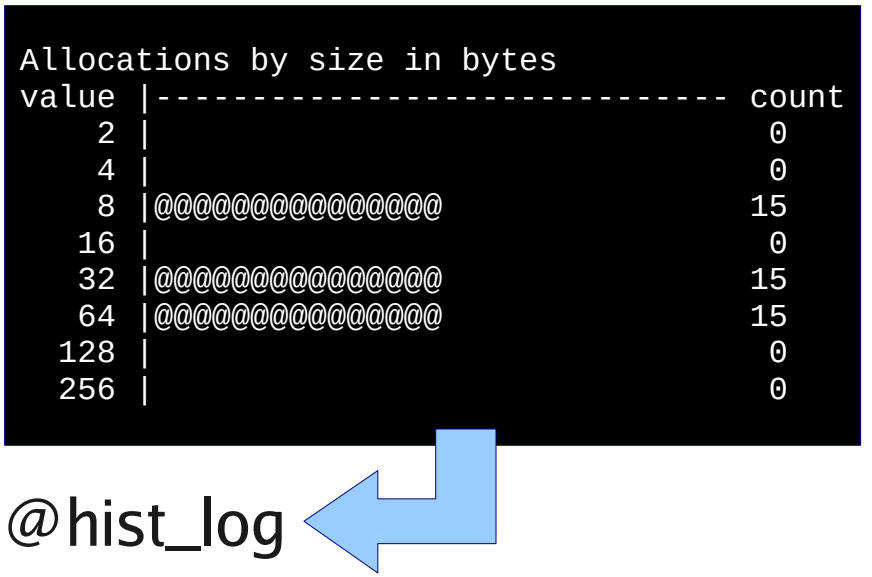
- Learn by looking at examples:
  - kprobe_example.c in linux-2.6/Documentation/kprobes.txt
  - SystemTap equivalent – kprobe_example.sh + regs.stp
- Compare the two different implementations
  - Complexity
  - Amount of code
  - Code safety
- Write less code yet achieve the same objective

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



fedora

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

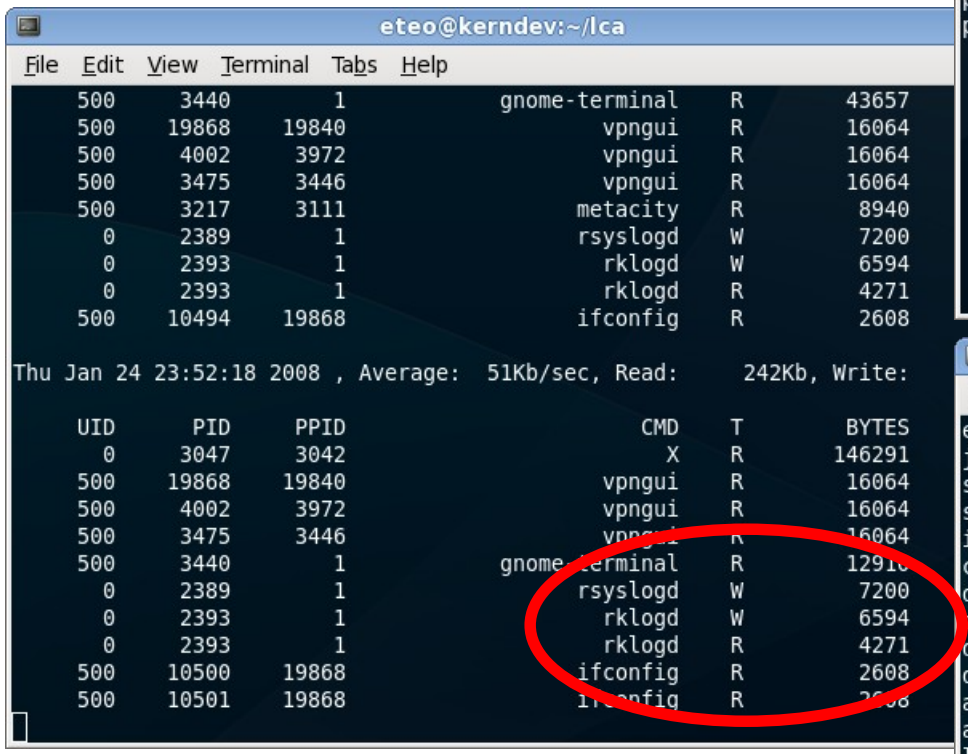
Some interesting (and useful) scripts

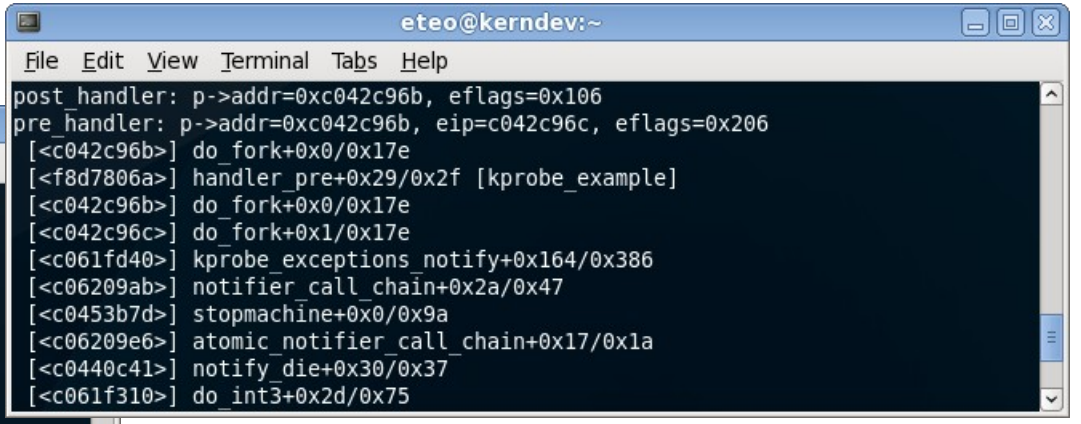
nettop.stp

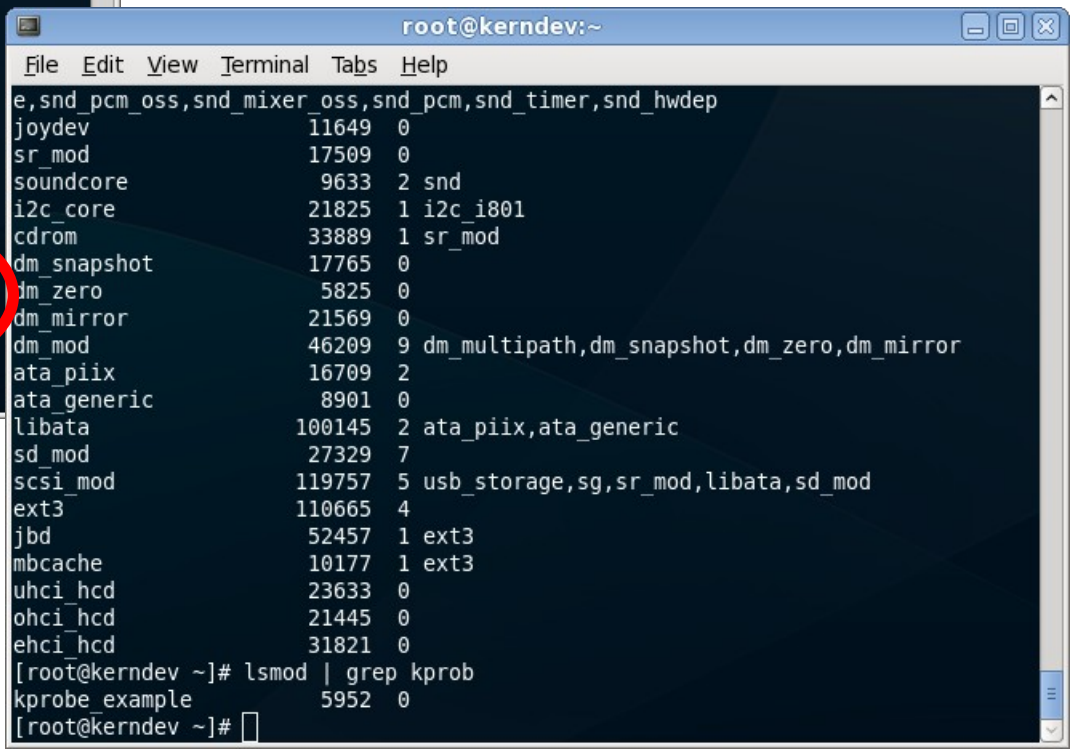
- Defined netdev.transmit and netdev.receive probes
- Prints network activity every 5 secs
- simpress sends and receives packets sometimes. Why?

iotop.stp

- Prints I/O activity







fedora

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
- My blog: http://www.kernel.sg/blog/category/systemtap/

Writing SystemTap Scripts

Eugene Teo

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