Kernel Debugging & Profiling

What to Expect?

- Kernel Debugging Tools & Techniques
- Kernel Profiling Tools & Mechanism
- Kernel Testing Ways

Debugging by Printing

- printk & dmesg
  - int klogctl(int cmd, char *bufp, int len);
- syslogd

  #include <syslog.h>

  void openlog(char *ident, int option, int facility);

  void syslog (int priority, char *format, ...);

  void closelog(void);
  - ident – typically short program name
  - option - LOG_CONS, LOG_PERROR, LOG_PID, ...
  - facility - LOG_AUTHPRIV, LOG_CRON, LOG_DAEMON, LOG_USER, ...
  - priority - LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_WARNING, LOG_NOTICE, LOG_INFO, LOG_DEBUG
- klogd – bridge between kernel ring buffer and syslogd

Debugging by Querying

- Through Kernel Windows
  - /proc
    - Process related stuff
    - General system information: cpuinfo, meminfo, ...
    - Drivers related info: devices, iomem, interrupts, ...
  - /sys (Utilities: sysfsutils, sysdiag)
    - Buses, Classes, Devices, ...
- Using ioctl
  - Through specific drivers
  - Using custom commands

Crash Dumping & Analysis

- Oops Analysis
  - CONFIG_BUG, CONFIG_KALLSYMS
  - Call Trace
  - cat /proc/kallsyms
  - objdump -d | -S <obj | exe>
- Crash Dumping (using 2 kernel images – could be same)
  - System Kernel: CONFIG_KEXEC, CONFIG_DEBUG_INFO
  - Dump-capture Kernel: CONFIG_CRASH_DUMP, CONFIG_PROC_VMCORE, Other Architecture specific CONFIG_s
  - Boot System Kernel with crashkernel=size@offset, or crashkernel=range:size
  - Overlay Dump-capture Kernel using kexec
  - After Crash (Simulated using SysRq+c, Or real)
    - System reboots w/ Dump-capture Kernel
  - Collect the dump using kdump, Or simply cp /proc/vmcore <dump file>
  - Analyse using <gdb | crash> vmlinux <dump file>

Kernel Hacking Related Options

- CONFIG_DEBUG_INFO (gcc -g)
- CONFIG_DEBUG_FS
- CONFIG_DYNAMIC_DEBUG
- CONFIG_DEBUG_STACKOVERFLOW
- CONFIG_KMEMCHECK
- CONFIG_DEBUG_KMEMLEAK, CONFIG_DEBUG_KMEMLEAK_EARLY_LOG_SIZE, ...
- CONFIG_LOCKUP_DETECTOR
- CONFIG_DEBUG_SPINLOCK, CONFIG_DEBUG_MUTEXES, ...
- CONFIG_MAGIC_SYSRQ
- CONFIG_EARLY_PRINTK
- CONFIG_DEBUG_SLAB, CONFIG_DEBUG_VM, ...
- CONFIG_DEBUG_STACK_USAGE
- CONFIG_PANIC_ON_OOPS
  - CONFIG_PANIC_ON_OOPS_VALUE

Kernel Debuggers

- Kernel Debugger (Frontend): kdb
  - CONFIG_DEBUG_INFO, CONFIG_MAGIC_SYSRQ
  - CONFIG_KGDB, CONFIG_KGDB_KDB, CONFIG_KGDB_SERIAL_CONSOLE
  - Basic Operations over Serial Console
  - Remote Debugging using
- kgdb (Kernel Debug Server)
  - Remote connection & debug through gdb vmlinux
- gdb <kernel src>/vmlinux /proc/kcore
  - Only to gather current kernel debug information

Miscellaneous Tools & Techniques

- Early Init Debugs
  - CONFIG_SERIAL_EARLYCON
  - CONFIG_EARLY_PRINTK (direct on h/w)
  - Light & Sound
  - JTAG
- Hardware Protocol Debug
  - Hardware Protocol Analyzers
- Network Debugging
  - Driver level: ethtool [options] <dev i/f>
  - Packet level: tcpdump [options, e.g. -vv] -i <dev i/f>

Kernel Probes

- kprobes → CONFIG_KPROBES
- jprobes → Specialized Kprobes
  - For probing function entry points
- kretprobes → Return Kprobes
  - For probing function exit points

Tracing

- Single Process Tracing using strace
  - Traces the system calls made by an application
  - Command: strace [options] <application>
  - Excellent options to tune to get only desired output
- Linux Trace Toolkit
  - Patch @ http://www.opersys.com/LTT
  - Core Module
  - Code using Trace Services
  - Daemon: tracedaemon
  - Utilities: tracereader, tracevisualizer
- Function Tracer
  - CONFIG_FUNCTION_TRACER & co
  - /sys/kernel/debug/tracing

Profiling & Code Coverage

- System Profiling
  - From /proc using LTTng
- Kernel Profiling: Oprofile
  - CONFIG_PROFILING, CONFIG_OPROFILE, CONFIG_APIC
  - Daemon: oprofiled
  - Utilities: opcontrol, opreport, op_help, ...
- Kernel Performance Profiling using perf
  - CONFIG_PERF_EVENTS & co
  - stat, list, top, record, report, ...
- Kernel Code Coverage:
  - CONFIG_GCOV_KERNEL, CONFIG_GCOV_PROFILE_ALL

Testing Possibilities

- Linux Test Project (LTP)
  - Hosted @ http://ltp.sourceforge.net
  - Suite of around 3000 tests
  - Exercises various parts of the Kernel
  - Mostly automated, except net & storage ones
  - Command: runltp -p -l logfile
- User Mode Linux (UML)
  - Hosted @ http://user-mode-linux.sourceforge.net
  - A Kernel Instance as a User Mode Process
  - Kernel Debugging without “oops”ing
  - Good for testing & experimenting with unstable kernels

What all have we learnt?

- Kernel Debugging Tools & Techniques
  - Debugging by Printing & Querying
  - Crash Dumping & Analysis
  - Kernel Hacking Options
  - Kernel Debuggers
  - Miscellaneous Tools & Techniques
  - Kernel Probes
  - Tracing
- Kernel Profiling Tools & Mechanism
- Kernel Testing Ways

Any Queries?