Crash Dump Analysis
System Debugging in Linux

Jakub Jermář
Martin Děcký
Overview

- **User space debugging tools**
  - Mostly already mentioned
    - gdb and friends
- **Kernel debugging tools**
  - Historically less officially supported tools (in vanilla kernel)
    - Different developer's attitude
    - printk() as the ultimate debugging tool
Linux Kernel Crash Dump

  - Kernel patches for simple crash dump generation on kernel panic
    - Supported only on SCSI devices
  - In-kernel debugger and post-mortem debugger
    - Similar features as `mdb -K`
      - Heuristical analysis
  - Active development during 2.5.x timeframe
    - Last supported release from 2006
Standard debugging means

- **Magic SysRq hot keys**
  - For dealing with hangs and security issues
    - Operator's intervention to the running system
    - Can be enabled/disabled by `/proc/sys/kernel/sysrq`
      - Alt + SysRq + 0 .. 9 set console logging level
      - Alt + SysRq + C activate kexec and generate a crash dump
      - Alt + SysRq + B immediate reboot
      - Alt + SysRq + O immediate shutdown
      - Alt + SysRq + S sync all mounted filesystems
      - Alt + SysRq + U remount all filesystems read-only
Standard debugging means (2)

- Alt + SysRq + P  dump registers to console
- Alt + SysRq + T  dump process information to console
- Alt + SysRq + L  dump stack traces of running threads
- Alt + SysRq + M  dump memory statistics to console
- Alt + SysRq + D  dump locked locks to console
- Alt + SysRq + K  kill all processes on the current console
- Alt + SysRq + E  terminate all processes except init
- Alt + SysRq + I  kill all processes except init
- Alt + SysRq + F  execute the OOM killer
- Alt + SysRq + N  reset nice level of all real-time processes
- Alt + SysRq + X  switch off raw keyboard mode
Standard debugging means (3)

- Activate from command line by writing into /proc/sysrq-trigger
- Activate over network by a special sysrqd server
  - Raising Elephants Is So Utterly Boring
  - Reboot Even If System Utterly Broken
    - Raw keyboard
    - Send SIGTERM to all processes
    - Send SIGKILL to all processes
    - Sync data to disk
    - Remount all filesystems read-only
    - Reboot
Standard debugging means (4)

- **kprobes**
  - Basic kernel instrumentation
    - An execution of any kernel function can trigger a callback function
- **kexec (kdump)**
  - Booting a new (Linux) kernel without physical reboot
    - Physical memory can be (more or less) retained and analyzed by the new kernel (as a crash dump)
User-Mode Linux

- **UML**
  - Special pseudo-hardware architecture
    - Otherwise compatible with the target architecture
    - Running Linux kernel as a user space process
      - Originally a virtualization effort
    - Great for debugging and kernel development
      - A plain standard gdb can be used to attach to the running kernel
      - Guest threads are threads of the UML process
        - Slightly more complicated to follow processes
KDB

- Linux kernel debugger
    - A set of patches to the vanilla kernel
      (very unprobably to be integrated soon)
  - For local debugging
  - Possibilities and features very similar to `mdb -K`
    - Activation in `/proc/sys/kernel/kdb`
    - Break-in by the *Pause* key (Ctrl+A on serial console)
      - `md` memory display
      - `mm` memory modify
KDB (2)

- rd  register dump
- rm  register modify
- bt  stack trace
- bp  set breakpoint

...
KGDB

- Kernel GDB
  - Since 2.6.26 integrated in the vanilla kernel (finally!)
    - Client/server debugger for remote kernel debugging
      - Over serial line or ethernet
      - Client: Plain gdb
        - Assembly and source line debugging
        - Symbol information from vmlinux binary on the client
    - Activation
      - After loading the I/O module and connecting to the server
      - Just after boot when `kgdbwait` kernel option is used
KGDB (2)

- Features
  - Memory and register analysis on kernel oops or panic
  - Limited single-stepping of the running kernel

% gdb ./vmlinux
(gdb) set remotebaud 115200
(gdb) target remote /dev/ttyS0

% gdb ./vmlinux
(gdb) target remote udp:192.168.1.1::6443