A Totally Different View
Under the Hood with Android & Linux

Jonathan Levin

http://Technologeeks.com

Ready your Droids

- It helps to follow along and try some hands-on
  - (this is, after all, a tutorial, and not just another lecture)
- If you have a real device – great
  - But advanced tracing/debugging does need root access
- At a minimum, fire up a KK emulator, and adb to it.

What this isn't

- An ADB shell primer
- A CLI how-to
- A native-level/NDK how-to
- A debugging primer for Dalvik/DDMS

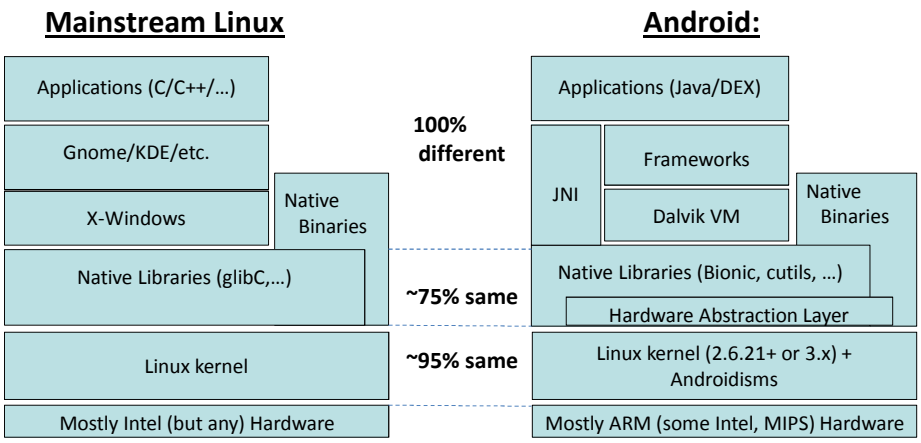
What this is

- Collection of native and CLI level debugging techniques
- Uses AOSP-supplied tools, and Linux facilities
- Applicable primarily to ARM, but also Intel and MIPS
- Actually also usable for Linux native level debugging
- An excerpt from my upcoming Android Internals book

The Book

- "Android Internals: A Confectioner's Cookbook"
- Unofficial sequel to "OS X and iOS Internals"
- To be released end of June, 2014
  - Along with Android Lollipop/Licorice/L* announcement?
- Updated for KK (4.4.2/API 19), with more to follow
- http://newandroidbook.com/
  - FAQ, TOC and plenty of bonus materials

Android Architecture



ADB and the shell

- ADB provides a command line shell as uid shell

```
shell@htc_m8wl:/ $ id
uid=2000(shell) gid=2000(shell) groups=1003(graphics),1004(input),1007(log),
1009(mount),1011(adb),1015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin),
3002(net_bt),3003(inet),3006(net_bw_stats) context=u:r:shell:s0
```

- Shell is MirBSD Korn shell, with scripting abilities
- Recommendation: Install SSHD (or dropbear, etc)
  - Frees you from tethering requirement, fully remote
  - Allows easier (and safer) root access
  - Will require public key authentication only (no password..)

ToolBox vs. BusyBox

- Most CLI commands implemented via toolbox
- Toolbox is Android specific subset of busybox
  - Pros: linked with Bionic, recognizes AIDs
  - Cons: limited toolset, partial functionality
- Recommendation: Install BusyBox
  - Statically linked binary, so no dependencies
  - Can compile from source, but plenty of binaries out there

Getting tools to your device

- A lot of the tools you need are right in the emulator
  - Most in /system/xbin, and not present in most devices
- Can adb pull from emulator image, push to device
- Android version should match
- Remember to move libraries as well!
  - Find dependencies using objdump –x | grep NEEDED

The procfs (/proc) filesystem

- A plethora of diagnostic information:
  - General system diagnostics (/proc root)
  - Subsystem information (/proc/bus, /proc/irq, …)
  - Sysctl variables (/proc/sys)
  - Per process diagnostics (/proc/[0-9]*])

The sysfs (/sys) filesystem

- Complements /proc, and provides:
  - Hardware and device representations
  - Kernel module information and parameters (/sys/module)
  - Kernel subsystem control

The debugfs (/d) filesystem

- Exclusively for kernel-level debugging and control
  - Kernel ftrace functionality
  - Binder debugging
  - Android atrace

```
root@Android:/ # mount -t debugfs none /sys/kernel/debug
root@Android:/ # mount | grep debug
none /sys/kernel/debug debugfs rw,relatime 0 0
```

- Usually mounted as /sys/kernel/debug, symlinked /d

keychords

- Little known feature of /init
- Binds services/commands to key combination
  - "keys" are physical buttons on device, as Android codes
- Uses /dev/keychord, where available
- Specify "keycodes" combination in /init.rc or other rc

Activity Diagnostics

- Tracing = monitoring run time activity of process
- Uses:
  - performance benchmarking
  - Logging and monitoring resource access

Activity Diagnostics - /proc

- A cornucopia of per process related information:

/proc entry	Provides
/proc/$pid/cwd	Symbolic link to current working directory
/proc/$pid/cmdline	NULL separated argv[] of process
/proc/$pid/fd	Directory with symbolic links to open descriptors
/proc/$pid/fdinfo	Information about open descriptors
/proc/$pid/status	Human readable general statistics (VM + More)
/proc/$pid/task	Directory per thread
/proc/$pid/wchan	Wait channel (indicates kernel syscall block/sleep)

Activity Diagnostics - /proc

- Iterating over task/* will show threads
- Threads largely have same stats, save:
  - Status/Name – Dalvik threads are named with prctl(2)
  - wchan – Kernel wait channel/syscall
- You can grep name $t/status to isolate threads
- You can then trace or suspend specific threads

Activity Diagnostics - Tools

- AOSP provides the lsof tool to list open files
  - Not just files, but actually any file descriptor for process
- Extremely useful with grep to isolate files

Activity Diagnostics - Tools

- AOSP also provides the strace binary to trace syscalls
  - Hands down, the #1 debugging tool out there
  - Based on ptrace(2) API, no dependencies
- Useful in oh-so-may ways:
  - Can actually parse and present system call arguments
  - Can follow forks and threads
  - Can be used for timing of syscalls
  - Can introduce artificial latency(!)

Activity Diagnostics - Tools

- The ltrace tool can also be ported to Android
  - Similar to ltrace, but provides **library call** information
  - Uses ptrace(2), but a lot heavier, and needs libelf.
- Supplements strace when your problem is in a lib:
  - Arguments and features similar to strace
  - Can also be used for syscalls (with –S)

Activity Diagnostics - Triggers

- Linux doesn't really support file access triggers
  - Inotify is an exception, but no shell command for it*
  - Still no notification for in-file access (say, certain offset)
- Using /proc and a little bit of scripting, however…

* - Well, there is ***now***.. Check http://NewAndroidBook.com/files/inotify.tar

Memory Diagnostics

- RAM is the most important resource in Android
- Applications leave in perpetual fear of OOM/LMK
- Most memory in Android is shared when possible
- Important to understand memory diagnostics

Low Memory Killer

- Protector of all droids, sworn adversary of all apps
  - Linux OOM is not-so-deterministic.
  - LMK more predictable – but more conservative
- Each process has an oom_score and an oom_adj
  - Native apps can cheat death – Dalvik ones can't
- LMK parameters can be tweaked through sysfs

/sysfs entry	Provides
…adj	Array of oom_adjs to kill on minfree hit
…minfree	Array of 4k multiples to start kill adj on
cost	Memory shrinker cost
debug_level	Verbosity for kill operations (1-5)

Memory Diagnostics

- VSS: Virtual Set Size (a.k.a VMSize)
- RSS: Resident Set Size
  USS = Unique Set Size + ShSS = Shared Set Size
  HWM = RSS Peak
- PSS: Proportional Set Size
  USS + (ShSS/#Shares)

Memory Diagnostics - /proc

- /proc filesystem provides key memory statistics:
  - Systemwide:

/proc entry	Provides
/proc/meminfo	Global memory statistics
/proc/vmstat	Kernel view of global memory, per variables

  - Per process:

/proc entry	Provides
/proc/$pid/maps	Address Space Layout (mmapped and anon)
/proc/$pid/smaps	As maps + per mapping information
/proc/$pid/status	VM Statistics, and much more

Memory Diagnostics - tools

- AOSP provides procrank and librank tools:
  - procrank: Ranks processes by memory utilization
  - librank: Ranks libraries by memory utilization (sharing)
- KitKat provides the memtrack tool
  - Logs memory utilization to Android logs

UpCall Scripts

- Android provides several CLI interfaces to Dalvik

Upcall tool	Provides
am	Interface to Activity Manager
bmgr	Backup Manager
content	Interface with Android Content Providers
ime	Input-Method-Editors
input	Interface with InputManager, inject events, etc
monkey	Stress/Fuzz test tool
pm	Interface to PackageManager
settings	Get/set system settings
svc	Control power, data, wifi and USB
wm	Interact with the Window Manager

The service tool

- CLI interface to servicemanger and the Binder
- Simple, but powerful
- Can automate virtually all Android services
  - Does require root access for some of the services
- Depends on service called, virtually undocumented*

* - Well, until ***now***.. That's why I wrote the book - http://NewAndroidBook.com/

Input Events

- Toolbox's getevent/sendevent can automate input
- The input upcall script can inject to input manager

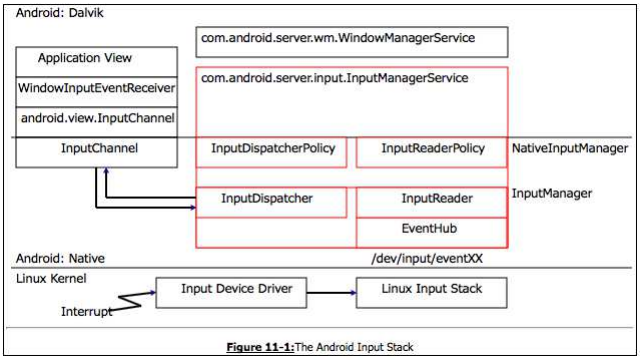


Figure 11-1: The Android Input Stack

Post Mortem Debugging

- Android doesn't support core dumps by default
  - Storage space is limited, and cores can be pretty big
  - ulimit –c 0 is set in /init (via setrlimit) and inherited
- Tombstones used instead of cores
  - Application crashes, debuggerd is notified
  - Checks if debug.db.uid property is set, to wait for gdb
  - Otherwise, engraves "tombstone" (crash report)

Tombstones

- Debuggerd uses Linux's ptrace(2) API to:
  - Enumerate all threads
  - Get register state for each thread
  - Get Stack trace for all threads
  - Get stack and instruction pointer memory contents
- Tombstone data is highly architecture specific

If you *do* want cores..

- Set ulimit to unlimited for crashing process
  - If you actively set to 0, must be root to unset
- Modify /proc/sys/kernel/core_pattern
  - Specify filename, can use %h, %e, %u, %p
  - Can also specify pipe (|) and command name
    - Command accepts core via stdin (e.g. HTC's dalvik_coredump.sh)