Android Debugging and Performance Analysis

CC-BY-SA 3.0 - Attribution requirements and misc., **PLEASE READ**:

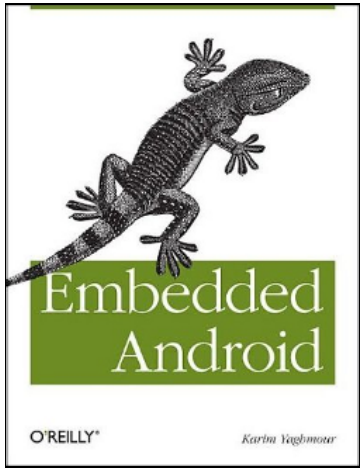
- This slide must remain as-is in this specific location (slide #1), everything else you are free to change; including the logo :-)
- Use of figures in other documents must feature the below "Originals at" URL immediately under that figure and the below copyright notice where appropriate.
- You are FORBIDDEN from using the default "About" slide as-is or any of its contents.

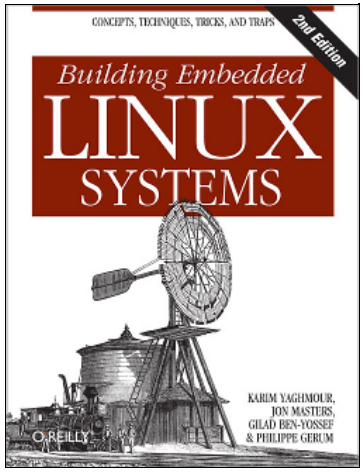
Copyright (C) 2013-2015, Opersys inc.

These slides created by: Karim Yaghmour

Originals at: **http://www.opersys.com/training/**

About





- Introduced Linux Trace Toolkit in 1999
- Originated Adeos and relayfs (kernel/relay.c)
- Training, Custom Dev, Consulting, ...

About this specific class

Android debugging is dirty business

Default AOSP builds have issues

… ergo …

Runtime adjustments will be made

Goals - High Level

- Understand the set of debugging and performance monitoring tools and capabilities available in Android
- Understand the internals and limitations of each
- Get hands-on experience with the tools and capabilities
- Determine which ones are most appropriate, useful and/or important for a given task

Goals - Specifics

- Understand the Android stack's debugging mechanisms and their internals
- Debug from the app level all the way down to kernel drivers
- Using Linux debugging tools with Android
- Learning about Android-specific tools
- Monitor performance and latencies
- Quantify and analyze memory usage
- Breakpoint and step through the stack
- Apply commonly-used techniques for framework debugging
- Familiarize with lesser-known tools and capabilities built into Android

HANDS ON

Prerequisites

- C/C++
- Java
- Linux command line
- Android internals
- Linux kernel internals
- Linux device drivers
- ARM architecture

Topics

1. Internals Architecture Quick Recap
2. Working with the AOSP Sources
3. Classifying and Analyzing Tools
4. Kernel Tools and Capabilities
5. Android-Agnostic User-Space Tools
6. Android-Specific User-Space Tools
7. Java Tools
8. System Services Interfacing
9. Other Tools and Techniques
10. glibc User-Space

Courseware

- These slides
- Exercises
- Online documentation

"Use the Source, Luke, use the Source. Be one with the code." -- Linus Torvalds

Hands-On Environment

Host

- Ubuntu-based system
- 50GB / AOSP

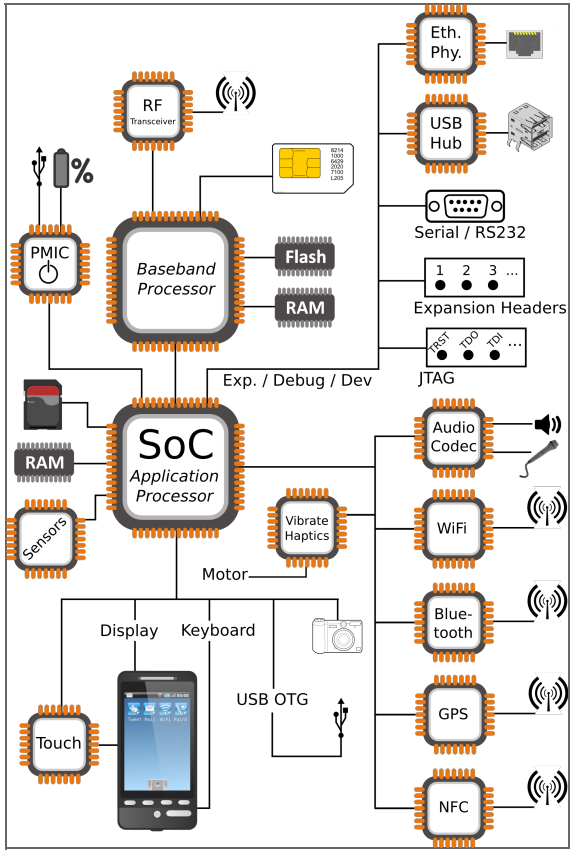
Target

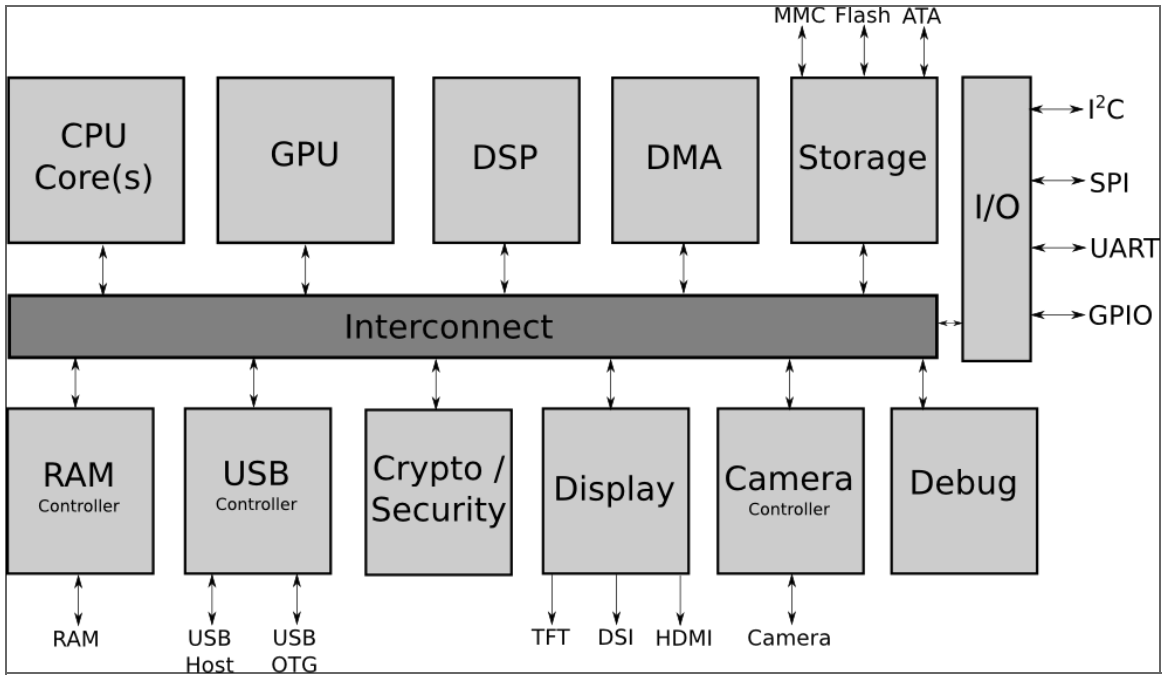
- Nexus 7 2013 ("flo")
- Qualcomm Snapdragon S4 Pro – APQ8064
- Krait CPU, 4-core, 1.51 GHz, 2MB L2 cache
- 2 GB on-board DDR3 (PCDDR 533MHz)
- 16 GB eMMC
- Combined power/usb

Internals Architecture Quick Recap

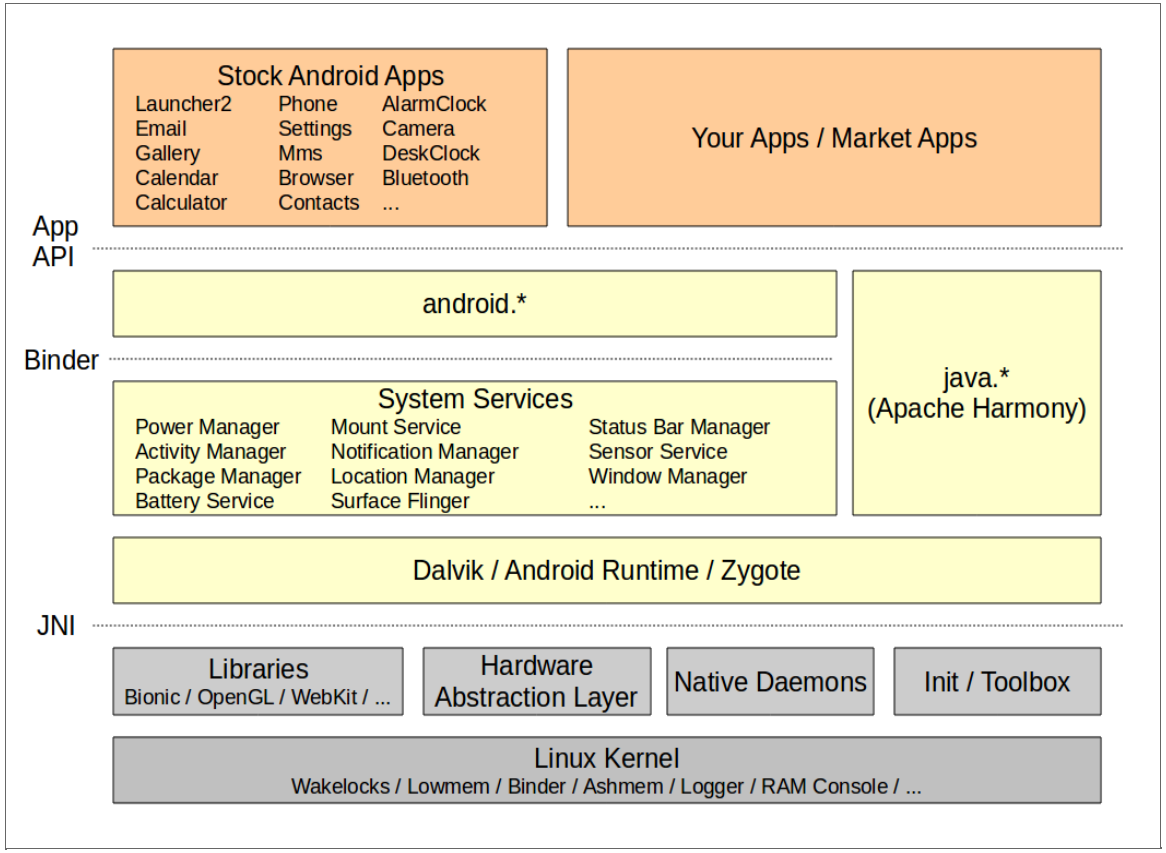
- Hardware used to run Android
- AOSP
- Binder
- System Services
- HAL
- Call walkthrough
- System startup
- Debug setup
- Network boot
- Symbolic debugging

1. Hardware used to run Android

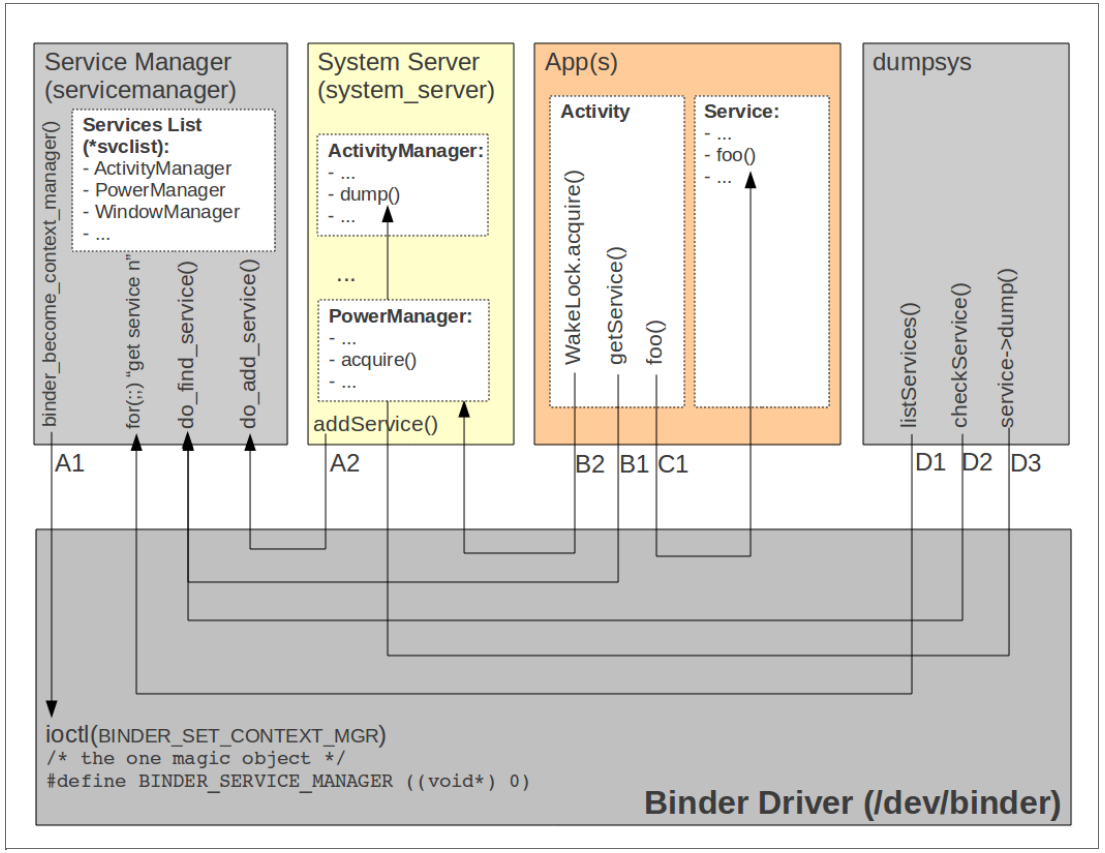



MMC
Flash
ATA
CPU Core(s)
GPU
DSP
DMA
Storage
I/O
I²C
SPI
UART
GPIO
Interconnect
RAM Controller
USB Controller
Crypto / Security
Display
Camera Controller
Debug
RAM
USB Host
USB OTG
TFT
DSI
HDMI
Camera

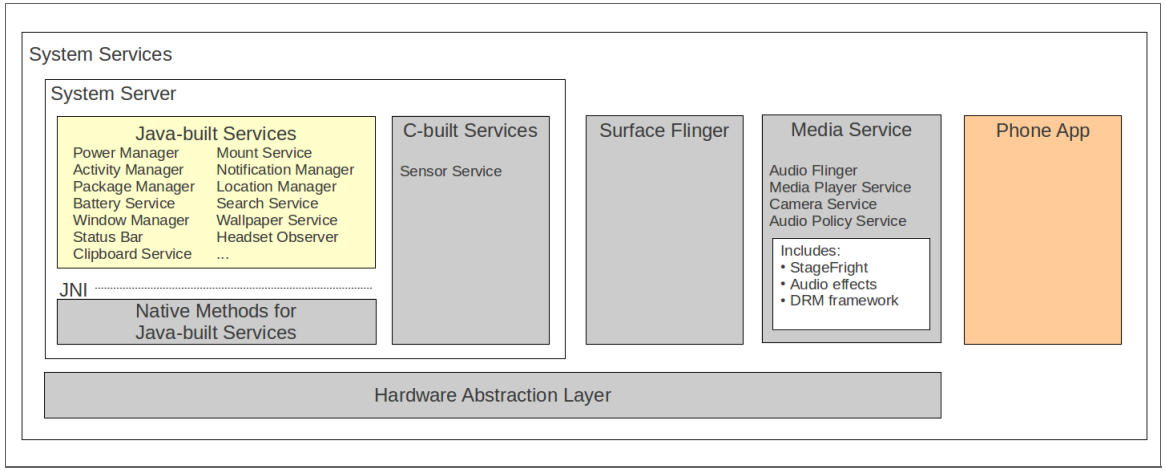
2. AOSP


Stock Android Apps
Launcher2
Email
Gallery
Calendar
Calculator
Phone
Settings
Mms
Browser
Contacts
AlarmClock
Camera
DeskClock
Bluetooth
...
Your Apps / Market Apps
App API
android.*
java.*
(Apache Harmony)
Binder
System Services
Power Manager
Activity Manager
Package Manager
Battery Service
Mount Service
Notification Manager
Location Manager
Surface Flinger
Status Bar Manager
Sensor Service
Window Manager
...
Dalvik / Android Runtime / Zygote
JNI
Libraries
Bionic / OpenGL / WebKit / ...
Hardware Abstraction Layer
Native Daemons
Init / Toolbox
Linux Kernel
Wakelocks / Lowmem / Binder / Ashmem / Logger / RAM Console / ...

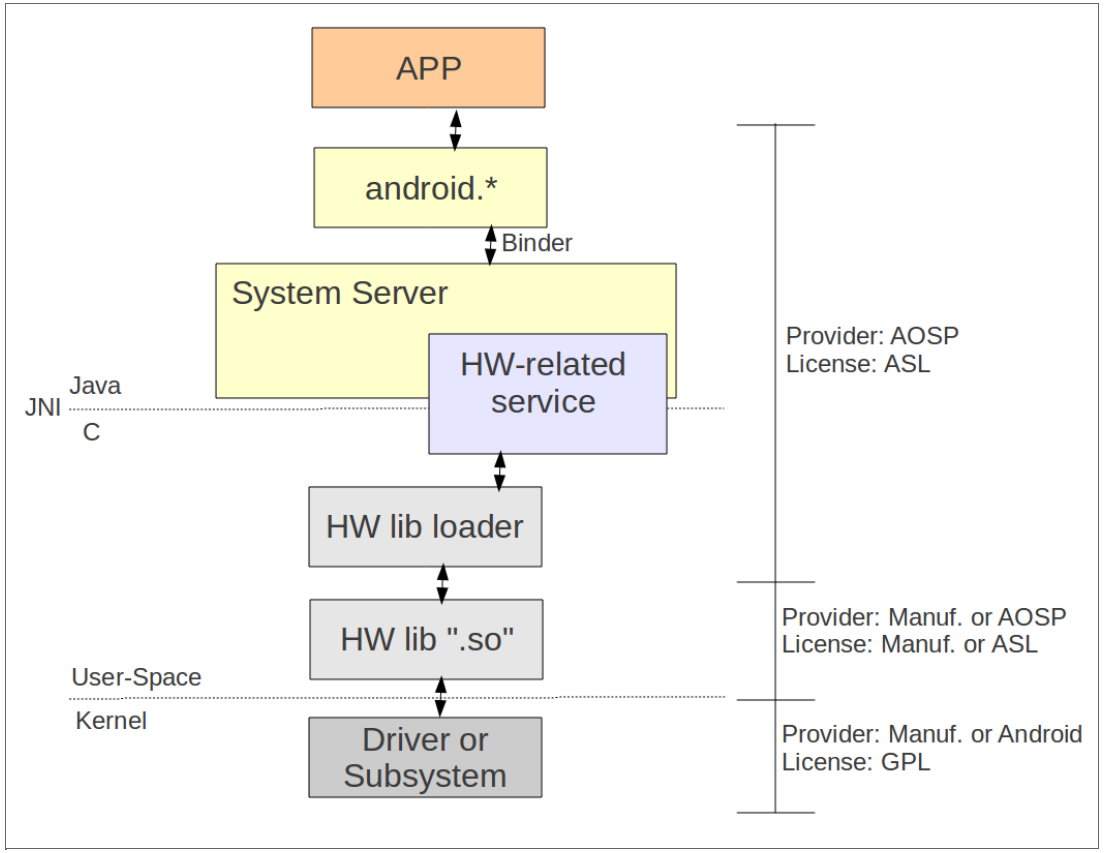
3. Binder


Service Manager
(servicemanager)
Services List
(*svclist):
- ActivityManager
- PowerManager
- WindowManager
- ...
binder_become_context_manager()
for(;;) "get service n"
do_find_service()
do_add_service()
System Server
(system_server)
ActivityManager:
- ...
- dump()
- ...
...
PowerManager:
- ...
- acquire()
- ...
addService()
App(s)
Activity
WakeLock.acquire()
getService()
foo()
Service:
- ...
- foo()
- ...
dumpsys
listServices()
checkService()
service->dump()
A1
A2
B2
B1
C1
D1
D2
D3
ioctl(BINDER_SET_CONTEXT_MGR)
/* the one magic object */
#define BINDER_SERVICE_MANAGER ((void*) 0)
Binder Driver (/dev/binder)

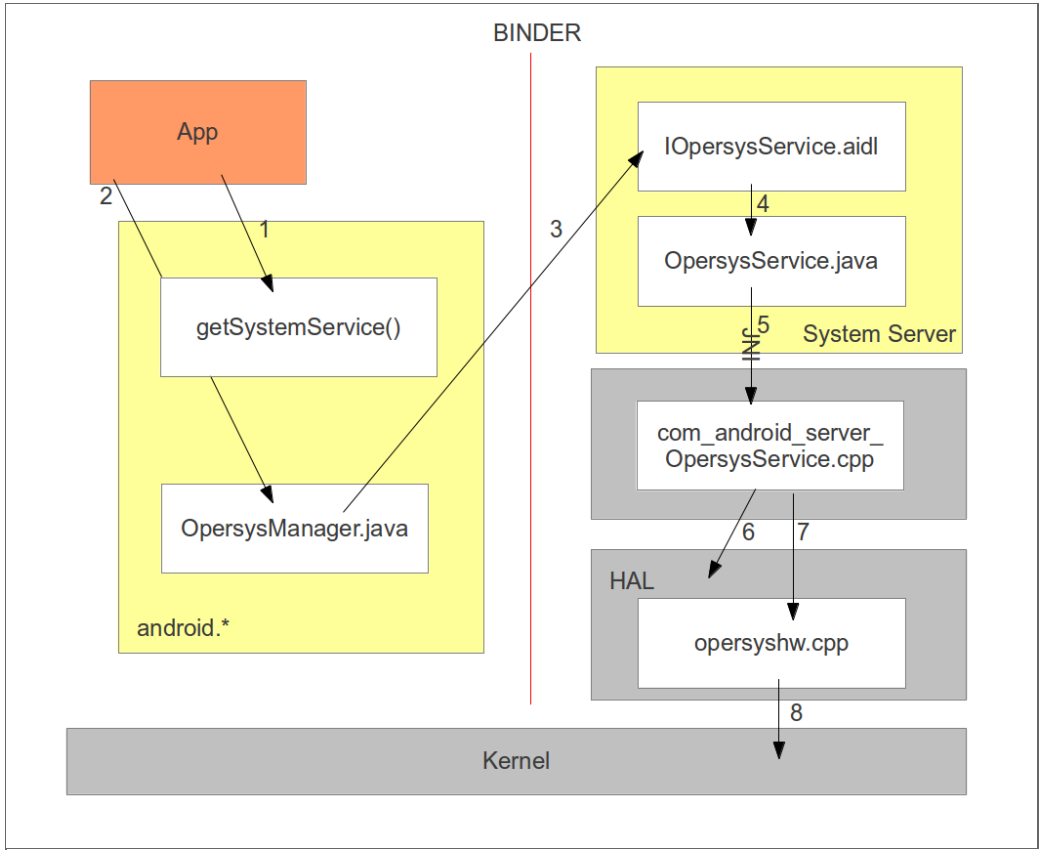
4. System Services



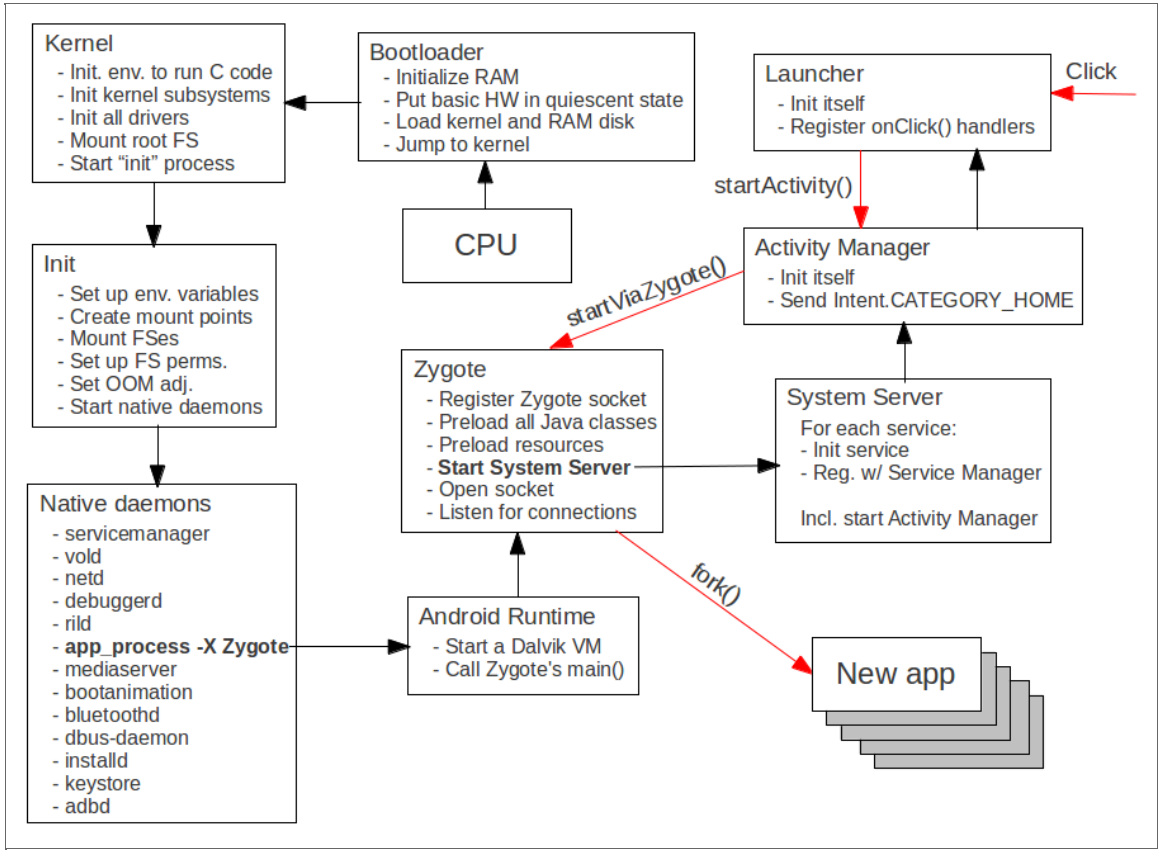
5. HAL


APP
android.*
Binder
System Server
HW-related service
Java
JNI
C
HW lib loader
HW lib ".so"
User-Space
Kernel
Driver or Subsystem
Provider: AOSP
License: ASL
Provider: Manuf. or AOSP
License: Manuf. or ASL
Provider: Manuf. or Android
License: GPL

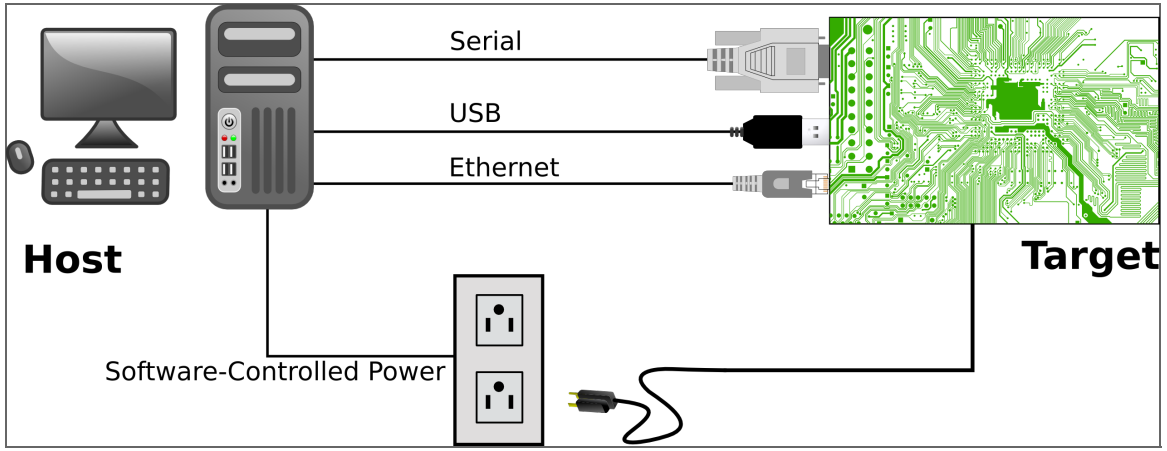
6. Call walkthrough



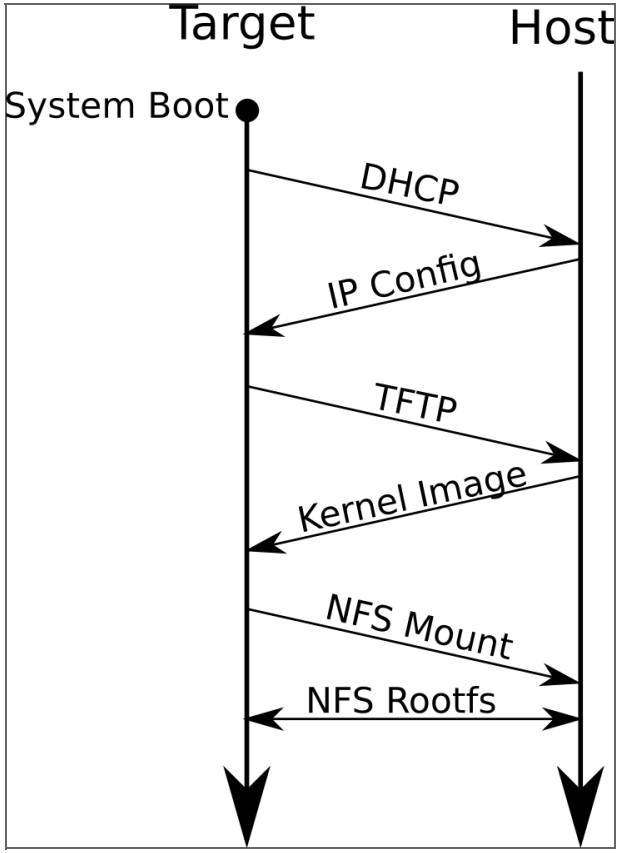
7. System startup



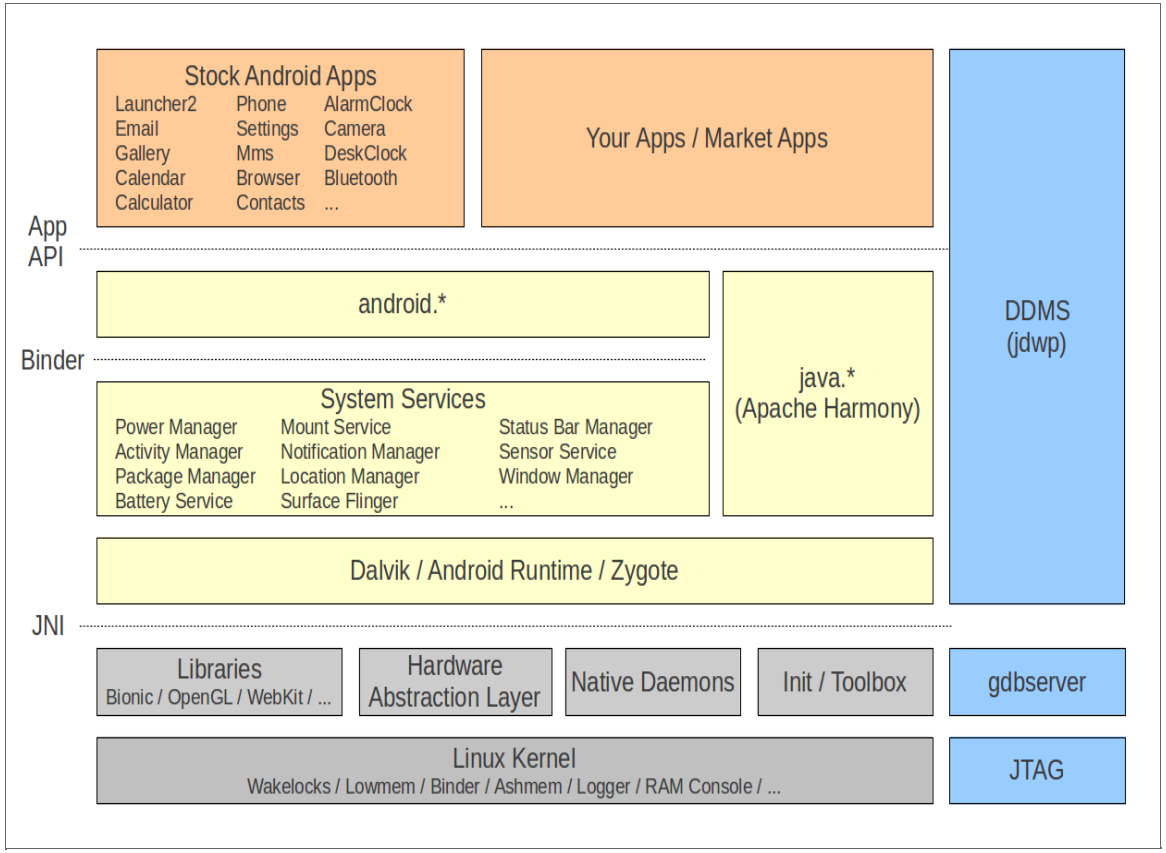
8. Debug setup



9. Network boot



10. Symbolic debugging



Working with the AOSP Sources

1. Basics
2. Preparing for Studio import
3. Importing into Studio
4. Browsing the sources

1. Basics

- repo
- build/envsetup.sh
  - godir
  - croot
  - mm
  - m
  - jgrep
  - cgrep
  - resgrep
- hmm
- lunch
- make -j8

2. Preparing for Studio import

AOSP:

- Get AOSP ... from Google or otherwise
- Extract if needed
- Configure, build, etc.

Android Studio:

- Get Android Studio from developer.android.com
- Extract
- Start and update and if needed

Create AOSP project files for Studio:

```
[aosp]$ make idegen && development/tools/idegen/idegen.sh
```

Sometimes you also need to fix an issue with "res.java":

```
[aosp]$ cd out/target/product/generic/obj/GYP/shared_intermediates
[aosp]$ mv res.java res.j && croot
```

3. Importing into Studio

- Start Studio:
  - Choose "Open an Existing Android Studio Project"
  - Select android.ipr form AOSP
  - Let it finish indexing
  - Close Studio
  - Restart Studio
  - Click on "Framework Detected" bubble

4. Browsing the Sources

- Right-click object type to be taken to declaration
- Browse classes through “Structure”
- Right-click "Find Usages"
- Toggle open files (Alt-left, Alt-right)
- Many other shortcuts, see:
  https://developer.android.com/sdk/installing/studio-tips.html
- Issues:
  - Can't compile with Studio ... still need “make”
  - For Java only

Classifying and Analyzing Tools

- Families of tools available in Linux/Android
- Use of sampling
- Use of software breakpoints
- Use of interrupts
- Ability to understand machine code
- Ability to read into internal data structures
- Use of statistics
- Use of buffering
- Time measurement
- Limitations
- Documentation vs. capabilities

Kernel Tools and Capabilities

1. Basic interfacing
2. Instrumentation
3. Analysis tools
4. Debugging

1. Basic Interfacing

- procfs
- sysfs
- configs
- debugfs
- dmesg/printk

1.1. procfs

- Mounted as `/proc` in Android
- Virtual filesystem maintained by kernel
- Traditionally the main way to expose internal info
- Since 2.6 it's meant for process info only
- All hardware-specific info is in sysfs instead
- `Documentation/filesystems/proc.txt`
- Used by a lot of tools: `ps`, `top`, `uptime`, etc.

```
1/              43/             60/             cpuinfo         net/
10/             45/             63/             crypto          pagetypeinfo
1007/           46/             64/             devices         partitions
1061/           47/             7/              diskstats       sched_debug
11/             48/             726/            dma-mappings    schedstat
12/             488/            741/            driver/         self/
13/             49/             756/            execdomains     slabinfo
14/             499/            773/            fb              softirqs
2/              5/              8/              filesystems     stat
25/             51/             835/            fs/             swaps
26/             512/            9/              interrupts      sys/
27/             52/             909/            iomem           sysrq-trigger
28/             523/            920/            ioports         sysvipc/
29/             53/             935/            irq/            timer_list
3/              54/             950/            kallsyms        tty/
30/             547/            971/            kmsg            uptime
31/             55/             988/            kpagecount      version
33/             56/             buddyinfo       kpageflags      vmallocinfo
34/             57/             bus/            loadavg         vmstat
362/            572/            cgroups         locks           yaffs
386/            58/             cmdline         meminfo         zoneinfo
39/             59/             config.gz       misc
40/             593/            consoles        mounts
413/            6/              cpu/            mtd
```

What's in here?

- Kernel-specific info:
  - memory information (`meminfo`)
  - verion/build (`version`)
  - CPU info (`cpuinfo`)
  - interrupt info (`irq/` and `interrupts`)
- One directory per PID:
  - memory maps (`maps`)
  - command line (`cmdline`)
  - `mem` file to access memory -- ptrace
  - sched stats (`sched`)
  - detailed process info (`status`)

A lot more stuff ...

1.2. sysfs

- Mounted as `/sys` in Android
- Virtual filesystem maintained by kernel
- Main way for kernel to publish its view of HW
- Tightly tied to kernel's device object model
- Enables hotplug functionality -- used by udev
- Allows user-space to write values to kernel-exposed properties
- `Documentation/filesystems/sysfs.txt`

```
block/      class/      devices/    fs/         module/
bus/        dev/        firmware/   kernel/     power/
```

1.3. configfs

- Would be mounted as `/config` if needed
- Not as often used as its counterparts
- Contrary to sysfs:
  - Enables user-space to create objects
- Used for configuring complex kernel-side subsystems:
  - USB composite devices
  - SCSI
- `Documentation/filesystems/configfs/configfs.txt`

1.4. debugfs

- Mount as `/sys/kernel/debug`
- Free "scratch area" for all things debugging
- No fixed rules of what can or has to be done
- Used by ftrace
- If you need to debug a driver, use this FS
- `Documentation/filesystems/debugfs.txt`

```
bdi/            hid/            sched_features  tracing/
binder/         memblock/       suspend_stats   wakeup_sources
```

1.5. dmesg/printk

- Meet the kernel's printf: `printk()`
- Defined: `include/linux/printk.h`

```
int printk(const char *fmt, ...);
```

- Implemented: `kernel/printk.c`
- Can loose data in cases of large output
- Widely-used throughout kernel sources
- Don't call while holding lock:
  - Has lock contention of its own

2. Instrumentation

- mcount
- tracepoints
- kprobes
- uprobes
- HW counters
- HW breakpoints

2.1. mcount

- `gcc`-based mechanism
- Trigger on `-pg` flag
- Originally-designed for `gprof`
- Kernel-side implemented in assembly:

```
arch/arm/kernel/entry-common.S
```

- Conditional to `CONFIG_FUNCTION_TRACER`
- Two possible behaviors -- `CONFIG_DYNAMIC_FTRACE`:
  - Hardcoded call
  - Dynamically-patched `nop`

2.2. Tracepoints

- Instrument your own code, for fun and profit
- In kernel:
  - Use built-in mechanism to define/use custom tracepoints
  - See

```
 kernel/tracepoint.c
include/linux/tracepoint.h
include/trace/*
include/trace/events/* -- definition of all global static tracepoints
Documentation/trace/tracepoints.txt
```

  - Example -- track context switches:
    - `include/trace/sched.h` uses this macro:

      TRACE_EVENT(sched_switch,...
    - This results in `trace_sched_switch()` to be created
    - `kernel/sched/core.c` uses this function

- Kernel instrumentation mechanism:
  - Conditional to `CONFIG_JUMP_LABEL`
  - If enabled, uses dynamically-patched `nop`s
  - If disabled, uses classic `if()`
  - Beware of `CONFIG_STRICT_MEMORY_RWX`
  - Probe using `register_trace_subsys_eventname()`

- In user-space:
  - Write to ftrace's buffer
  - That's what Android's atrace functionality does
  - `/sys/kernel/debug/tracing/trace_marker`
  - It's just a file
  - `open()`, `write()`, `write()`, `write()`, ...
  - Read your events as part of ftrace's output

2.3. kprobes

- Formal mechanism for dynamically adding probe points
- In mainline kernel since 2005:
  - Stems from IBM's previous work on DProbes
  - Trimmed-down version of DProbes functionality
- Requires module insertion
- Module must know insertion address/symbol
- 3 types of probes:
  - Kprobe => `register_kprobe()`
  - Jprobe => `register_jprobe()`
  - Kretprobe => `register_kretprobe()`
- Typically:
  - `module_init()` registers + provides handlers
  - `module_exit()` unregisters
- `Documentation/kprobes.txt`

What's a kprobe?

- Acts like a typical breakpoint
- Original instruction at destination is copied
- Breakpoint is inserted
- On hit, kprobe-registered `pre_handler` callback notified
- Copied instruction is single-stepped
- Then, kprobe-registered `post_handler` callback notified
- Execution continues at the next instruction
- Example: `samples/kprobes/kprobe_example.c`

What's a jprobe?

- It's a kprobe inserted at function entry-point
- Allows handler to inspect function's arguments
- Called function's stack is copied for inspection:
  - Only `MAX_STACK_SIZE` is copied -- 64 bytes on ARM
- Registered handler is called
- Copied stack is recopied over the original
  - gccism, see doc
- Example: `samples/kprobes/jprobe_example.c`

What's a kretprobe?

- Allows you to monitor function entry and exit
- kprobe inserted at function entry-point
- Return address is saved and replaced with handler
- Then `entry_handler` is called
- Function continues
- When function returns, return handler (`handler`) is called
- Example: `samples/kprobes/kretprobe_example.c`

Android support

- It's orthogonal to Android
- Kernel mechanism
- No user-space component
- No need for explicit Android support

Resources

https://lwn.net/Articles/132196/

http://www.linuxforu.com/2011/04/kernel-debugging-using-kprobe-and-jprobe/

https://sourceware.org/systemtap/kprobes/

2.4. uprobes

- User-space equivalent to kprobes
- Currently:
  - x86
  - PowerPC
  - ARM support coming in 3.15
    - Presently patch available from Linaro
- See:

```
kernel/events/uprobes.c
kernel/trace/trace_uprobe.c
Documentation/trace/uprobetracer.txt
```

2.5. HW counters

- Count key HW events without SW support
- Very HW-specific:
  - Arch-specific
  - CPU-specific
- Handled by `perf`, for better or worse
- `perf` designed to measure on overflow

2.6. HW breakpoints

- Create `breakpoints` on memory access
- Core is also handled by `perf`
- `kernel/events/hw_breakpoint.c`
- `samples/hw_breakpoint/data_breakpoint.c`

3. Analysis Tools

- SystemTap
- ktap
- BPF trace
- ftrace
- LTTng
- oprofile
- perf

3.1. SystemTap

- Problem:
  kprobes requires hand-crafted modules, for each probe point
- Need:
  Higher-level mechanism for defining and handling probe points
- Solution:
  SystemTap
- Built on kprobe mechanism
- External project from the kernel (IBM, RedHat, Intel)
- Effectively deprecates DProbes
- Full-fledged scripting language for creating/handling probes
- **HUGE** number of canned scripts for all sorts of uses
- **https://sourceware.org/systemtap/**

Android support

- None officially -- not in AOSP
- Maybe?:
  https://github.com/flipreverse/systemtap-android
- Also: requires a compiler to build the modules ...
- See here for a good discussion of the issues ... and a diagram:
  http://omappedia.org/wiki/Systemtap#Systemtap_and_Cross_Compilation

Resources

https://sourceware.org/systemtap/wiki

https://sourceware.org/systemtap/tutorial/

https://sourceware.org/systemtap/tapsets/

3.2. ktap

- Problem:
  - SystemTap requires a compiler
  - SystemTap requires loading modules
- Need:
  - Something similar to SystemTap, minus its issues
- Solution:
  - ktap
- Compiles scripts into bytecode
- Bytecode is interpreted by lua-based VM in kernel:
  - Seriously, it sounds scarier than it actually is
- Released in May 2013
- Initially positive feedback from key kernel developers
- Nack'ed by Ingo Molnar
- Aims to be the "DTrace" of Linux

Android support

- None that I know of, this is too new at this point (Jan 2014)
- Developer has embedded background so maybe ... just maybe
- Makefile doesn't seem to have "CROSS_COMPILE" prefix

Resources

http://www.ktap.org/

http://events.linuxfoundation.org/sites/events/files/lcjpcojp13_zhangwei.pdf

https://github.com/ktap/ktap

https://lwn.net/Articles/531059/

3.3. BPF

- "Berkeley Packet Filter"
- Bytecode for packet filtering
- In-kernel AOT/JIT
- Reuse for tracing filters
- About the BPF patches:
  - **https://lwn.net/Articles/593476/**
  - **https://lwn.net/Articles/593476/#internals**
- The BPF tracing filters:
  - **https://lwn.net/Articles/575531/**
- BPF gcc-to-bpf user-space backend:
  - **https://github.com/iovisor/bpf_gcc/commit/9e7223f8f09c822ecc6e18309e89a574a23dbf63**

3.4. ftrace

- Kernel function **and** event tracer
- Relies on:
  - gcc's "-pg" flag (i.e. `mcount()`)
  - Tracepoints
- `/sys/kernel/debug/tracing/`

```
README                  options/                trace_options
available_events        per_cpu/                trace_pipe
available_tracers       printk_formats          tracing_cpumask
buffer_size_kb          saved_cmdlines          tracing_enabled
buffer_total_size_kb    set_event               tracing_max_latency
current_tracer          trace                   tracing_on
events/                 trace_clock             tracing_thresh
free_buffer             trace_marker
```

- `Documentation/tracing/ftrace.txt`

Kernel configuration options to watch for:

- CONFIG_FTRACE
- CONFIG_FUNCTION_TRACER
- CONFIG_FUNCTION_GRAPH_TRACER
- CONFIG_STACK_TRACER
- CONFIG_DYNAMIC_FTRACE

Implementation - `kernel/trace/`

```
blktrace.c                trace_events.c              trace_output.h
ftrace.c                  trace_events_filter.c       trace_printk.c
Kconfig                   trace_events_filter_test.h  trace_probe.c
Makefile                  trace_export.c              trace_probe.h
power-traces.c            trace_functions.c           trace_sched_switch.c
ring_buffer_benchmark.c   trace_functions_graph.c     trace_sched_wakeup.c
ring_buffer.c             trace.h                     trace_selftest.c
rpm-traces.c              trace_irqsoff.c             trace_selftest_dynamic.c
trace_branch.c            trace_kdb.c                 trace_stack.c
trace.c                   trace_kprobe.c              trace_stat.c
trace_clock.c             trace_mmiotrace.c           trace_stat.h
trace_entries.h           trace_nop.c                 trace_syscalls.c
trace_event_perf.c        trace_output.c              trace_uprobe.c
```

Check if tracing is on:

```
# cat tracing_on
0
```

Check which tracers are available:

```
# cat available_tracers
blk function_graph wakeup_rt wakeup function nop
```

Check the current tracer:

```
# cat current_tracer
nop
```

Set the current tracer:

```
# echo function > current_tracer
```

Enable tracing -- beware the space with ">":

```
# echo 1 > tracing_on
```

The raw events:

```
# ls events/*
```

Check the content of a trace:

```
 # cat trace
# tracer: function
#
# entries-in-buffer/entries-written: 60028/3128571   #P:1
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
            adbd-55    [000] ....  1075.680000: __schedule <-schedule
            adbd-55    [000] ....  1075.680000: rcu_sched_qs <-__schedule
            adbd-55    [000] d...  1075.680000: deactivate_task <-__schedule
            adbd-55    [000] d...  1075.680000: dequeue_task <-deactivate_task
            adbd-55    [000] d...  1075.680000: update_rq_clock <-dequeue_task
            adbd-55    [000] d...  1075.680000: dequeue_task_fair <-dequeue_task
            adbd-55    [000] d...  1075.680000: update_curr <-dequeue_task_fair
            adbd-55    [000] d...  1075.680000: clear_buddies <-dequeue_task_fair
            adbd-55    [000] d...  1075.680000: account_entity_dequeue <-dequeue_task_fair
            ...
```

Stop tracing

```
# echo 0 > tracing_on
```

Clear a trace:

```
# echo > trace
```

Check buffer size:

```
# cat buffer_size_kb
1408
```

Set buffer size:

```
# echo 2048 > buffer_size_kb
```

Use function graph tracer:

```
# echo function_graph > current_tracer
```

Restart tracing:

```
# echo 1 > tracing_on
```

Check graph tracer output:

```
 # cat trace
# tracer: function_graph
#
# CPU  DURATION                  FUNCTION CALLS
# |     |   |                     |   |   |   |
 0)   0.000 us    |              } /* __sync_icache_dcache */
 0)   0.000 us    |              __sync_icache_dcache();
 0)   0.000 us    |              vm_normal_page();
 0)   0.000 us    |              __sync_icache_dcache();
 0)   0.000 us    |              _cond_resched();
 0)   0.000 us    |            } /* copy_pte_range */
 0)   0.000 us    |          } /* copy_page_range */
 0)               |          cap_vm_enough_memory() {
 0)   0.000 us    |            cap_capable();
 0)   0.000 us    |            __vm_enough_memory();
 0)   0.000 us    |          }
 0)   0.000 us    |          kmem_cache_alloc();
 0)               |          anon_vma_fork() {
 0)               |            anon_vma_clone() {
 0)   0.000 us    |              kmem_cache_alloc();
 0)   0.000 us    |              mutex_lock();
 0)   0.000 us    |              anon_vma_chain_link();
 0)   0.000 us    |              mutex_unlock();
 0)   0.000 us    |            }
...
```

Linux toolset

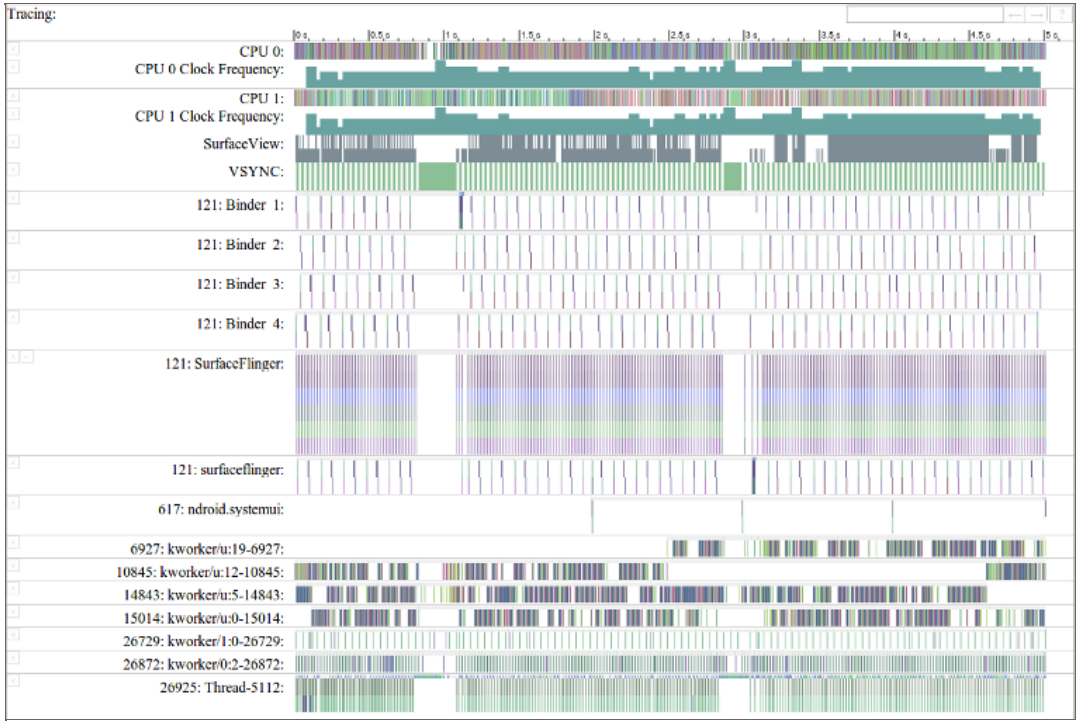
- `trace-cmd` (uses `splice()`)-- not avail. in Android
  http://git.kernel.org/cgit/linux/kernel/git/rostedt/trace-cmd.git
- KernelShark -- not avail. in Android
  http://people.redhat.com/srostedt/kernelshark/HTML/
- Neither of these are included in the AOSP

Support in Android

- Since 4.1 -- significant changes in 4.2
- ... **finicky** ...
- Android stack feeds events into ftrace
- Same entries in `/sys/kernel/debug/tracing`
- Neither `trace-cmd` nor KernelShark
- Android tools:
  - On the device: `atrace` -- native binary
  - On the host: `systrace` -- Python script
- `systrace` calls `atrace` over `adb`
- `systrace`-generated traces viewable w/ Chrome
  - And nothing but Chrome ... NIH?

```
 # atrace --help
usage: atrace [options] [categories...]
options include:
  -a appname      enable app-level tracing for a comma separated list of cmdlines
  -b N            use a trace buffer size of N KB
  -c              trace into a circular buffer
  -k fname,...    trace the listed kernel functions
  -n              ignore signals
  -s N            sleep for N seconds before tracing [default 0]
  -t N            trace for N seconds [defualt 5]
  -z              compress the trace dump
  --async_start   start circular trace and return immediatly
  --async_dump    dump the current contents of circular trace buffer
  --async_stop    stop tracing and dump the current contents of circular
                    trace buffer
  --list_categories
                  list the available tracing categories
```

```
root@flo:/ # atrace --list_categories
        gfx - Graphics
      input - Input
       view - View System
    webview - WebView
         wm - Window Manager
         am - Activity Manager
       sync - Sync Manager
      audio - Audio
      video - Video
     camera - Camera
        hal - Hardware Modules
        app - Application
        res - Resource Loading
     dalvik - Dalvik VM
         rs - RenderScript
     bionic - Bionic C Library
      power - Power Management
      sched - CPU Scheduling
        irq - IRQ Events
       freq - CPU Frequency
       idle - CPU Idle
       load - CPU Load
       sync - Synchronization
      workq - Kernel Workqueues
 memreclaim - Kernel Memory Reclaim
```


Tracing:
CPU 0:
CPU 0 Clock Frequency:
CPU 1:
CPU 1 Clock Frequency:
SurfaceView:
VSYNC:
121: Binder 1:
121: Binder 2:
121: Binder 3:
121: Binder 4:
121: SurfaceFlinger:
121: surfaceflinger:
617: ndroid.systemui:
6927: kworker/u:19-6927:
10845: kworker/u:12-10845:
14843: kworker/u:5-14843:
15014: kworker/u:0-15014:
26729: kworker/1:0-26729:
26872: kworker/0:2-26872:
26925: Thread-5112:

Google's doc:
https://developer.android.com/tools/help/systrace.html
https://developer.android.com/tools/debugging/systrace.html

Also have a look at these:

```
 /external/chromium-trace/systrace.py
/frameworks/native/cmds/atrace
/frameworks/base/core/java/android/os/Trace.java
/frameworks/base/core/jni/android_os_Trace.cpp
/frameworks/native/include/utils/Trace.h
/system/core/include/cutils/trace.h
/system/core/libcutils/trace.c
/frameworks/native/libs/utils/Trace.cpp
```

Look for:

- `ATRACE*` in c/cpp files
- `Trace.traceBegin()`/`trace.traceEnd()` in Java files

- Use in C files in 4.4:

```
 #include <cutils/trace.h>
...
#define ATRACE_TAG ATRACE_TAG_ALWAYS
...
ATRACE_BEGIN()
ATRACE_END()
```

- Use in C++ files -- you can also use ATRACE_CALL():

```
 #include <utils/Trace.h>
...
#define ATRACE_TAG ATRACE_TAG_ALWAYS
...
ATRACE_CALL()
```

Gotchas:

- Enabling on the command line:

```
# setprop debug.atrace.tags.enableflags ...
```

- Make sure the trace marker file is writeable (/sys/kernel/debug/tracing/trace_marker):
  - Either mount debugfs at startup
  - Or:

```
# chmod 222 /sys/kernel/debug/tracing/trace_marker
```

Use in drivers

- In the long-term:
  - Create your own events with `TRACE_EVENT()` macro
- For short-term debugging/instrumentation:
  - Use `trace_printk()`
  - `trace_printk()` is `EXPORT_SYMBOL_GPL()`'ed
  - Make sure your module is GPL-licensed:

    MODULE_LICENSE("GPL");
  - Otherwise, symbol will be unresolvable at load time

Resources:

https://lwn.net/Articles/365835/

https://lwn.net/Articles/366796/

https://lwn.net/Articles/370423/

http://elinux.org/Ftrace

3.5. LTTng

- Complete rewrite of the Linux Trace Toolkit
- Extremely scalable, low-overhead
- Very effective user-space tracing (UST)
- Mostly maintained out of tree:
  - Loadable module
  - Relies on existing kernel tracepoint functionality
- **http://lttng.org/**
- Very powerful visualization tools
- No "official" support for or in Android:
  - Scattered patches for Android support
  - User-space tracing requires SHM
- `trace_marker`-like functionality upstreamed

Resources:

https://lwn.net/Articles/491510/

https://lwn.net/Articles/492296/

3.6. oprofile

- System profiler: both kernel and user-space
- Originally based on system timer
- Relies on performance counters:
  - Most recently as provided by `perf`
- Must be disabled for `perf` to work
- AOSP has oprofile tools in `external/`:

```
 # opcontrol --help

opcontrol: usage:
   --list-events    list event types
   --help           this message
   --verbose        show extra status
   --verbose-log=lvl set daemon logging verbosity during setup
                    levels are: all,sfile,arcs,samples,module,misc
   --setup          setup directories
...
```

- `perf` seems to be favored these days
- **http://oprofile.sourceforge.net**

3.7. perf

- Initial goal: formal interface for performance counters
  - oprofile used its own custom/external module for those
- Now spans a lot events than just PMU-based
- Being pushed by fairly influential kernel developers
- Poorly documented
- Steep learning curve
- Great for statistical analysis, not for detailed tracing
- Counters saved on context switch, if per-process
- Works great on x86
- Underwhelming support for ARM SoCs
  - Actually works on Qualcomm SoCs ... with some elbow grease
- Implemented in `kernel/events/` and `tools/perf`

- Documentation:
  - `tools/perf/design.txt`
  - `tools/perf/Documentation/`
  - **https://perf.wiki.kernel.org/index.php/Tutorial**
- **MUST READ: "multiplexing and scaling events" in tutorial**:
  - Trying to monitor more events than there PMU counters will result in multiplexing and scaling of data collection
- There's a `perf` system call:

```
int sys_perf_event_open(struct perf_event_attr *hw_event_uptr,
                        pid_t pid, int cpu, int group_fd,
                        unsigned long flags);
```

- Requires CONFIG_PERF_EVENTS
- Unlike ftrace, really can't be used without `perf` command

```
# perf

usage: perf [--version] [--help] COMMAND [ARGS]

The most commonly used perf commands are:
  annotate        Read perf.data (created by perf record) and display annotated code
  archive         Create archive with object files with build-ids found in perf.data file
  bench           General framework for benchmark suites
  buildid-cache   Manage build-id cache.
  buildid-list    List the buildids in a perf.data file
  diff            Read two perf.data files and display the differential profile
  evlist          List the event names in a perf.data file
  inject          Filter to augment the events stream with additional information
  kmem            Tool to trace/measure kernel memory(slab) properties
  kvm             Tool to trace/measure kvm guest os
  list            List all symbolic event types
  lock            Analyze lock events
  probe           Define new dynamic tracepoints
  record          Run a command and record its profile into perf.data
  report          Read perf.data (created by perf record) and display the profile
  sched           Tool to trace/measure scheduler properties (latencies)
  script          Read perf.data (created by perf record) and display trace output
  stat            Run a command and gather performance counter statistics
  test            Runs sanity tests.
  timechart       Tool to visualize total system behavior during a workload
  top             System profiling tool.

See 'perf help COMMAND' for more information on a specific command.
```

Get basic stats:

```
# perf stat -a sleep 5

Performance counter stats for 'sleep 5':

       5014.375095 task-clock                #    1.000 CPUs utilized
              371 context-switches          #    0.000 M/sec
                0 CPU-migrations            #    0.000 M/sec
              270 page-faults               #    0.000 M/sec
         49315140 cycles                    #    0.010 GHz                      [80.45%]
                0 stalled-cycles-frontend   #    0.00% frontend cycles idle     [78.46%]
                0 stalled-cycles-backend    #    0.00% backend  cycles idle     [96.02%]
         16766094 instructions              #    0.34  insns per cycle          [72.67%]
          1826454 branches                  #    0.364 M/sec                    [76.27%]
           158411 branch-misses             #    8.67% of all branches          [76.58%]

       5.013001679 seconds time elapsed
```

Monitor what functions are using the CPU:

```
    PerfTop:     935 irqs/sec  kernel:91.6%  exact:  0.0% [1000Hz cycles],  (all, 1 CPU)
-------------------------------------------------------------------------------

             samples  pcnt function               DSO
             _______ _____ _____________________ _____________________

               13.00 34.2% dvmAsmInstructionStart /system/lib/libdvm.so
                6.00 15.8% strcmp                 /system/lib/libc.so
                6.00 15.8% __vfprintf             /system/lib/libc.so
                6.00 15.8% dlmalloc               /system/lib/libc.so
                5.00 13.2% dvmJitToInterpNoChain  /system/lib/libdvm.so
...
```

See the events it can monitor:

```
  # perf list

List of pre-defined events (to be used in -e):
  cpu-cycles OR cycles                               [Hardware event]
  stalled-cycles-frontend OR idle-cycles-frontend    [Hardware event]
  stalled-cycles-backend OR idle-cycles-backend      [Hardware event]
  instructions                                       [Hardware event]
  cache-references                                   [Hardware event]
  cache-misses                                       [Hardware event]
  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  bus-cycles                                         [Hardware event]

  cpu-clock                                          [Software event]
  task-clock                                         [Software event]
  page-faults OR faults                              [Software event]
  minor-faults                                       [Software event]
  major-faults                                       [Software event]
  context-switches OR cs                             [Software event]
  cpu-migrations OR migrations                       [Software event]
  alignment-faults                                   [Software event]
  emulation-faults                                   [Software event]
```

continued:

```
  L1-dcache-loads                                  [Hardware cache event]
  L1-dcache-load-misses                            [Hardware cache event]
  L1-dcache-stores                                 [Hardware cache event]
  L1-dcache-store-misses                           [Hardware cache event]
  L1-dcache-prefetches                             [Hardware cache event]
  L1-dcache-prefetch-misses                        [Hardware cache event]
  L1-icache-loads                                  [Hardware cache event]
  L1-icache-load-misses                            [Hardware cache event]
  L1-icache-prefetches                             [Hardware cache event]
  L1-icache-prefetch-misses                        [Hardware cache event]
  LLC-loads                                        [Hardware cache event]
  LLC-load-misses                                  [Hardware cache event]
  LLC-stores                                       [Hardware cache event]
  LLC-store-misses                                 [Hardware cache event]
  LLC-prefetches                                   [Hardware cache event]
  LLC-prefetch-misses                              [Hardware cache event]
  dTLB-loads                                       [Hardware cache event]
  dTLB-load-misses                                 [Hardware cache event]
  dTLB-stores                                      [Hardware cache event]
  dTLB-store-misses                                [Hardware cache event]
  dTLB-prefetches                                  [Hardware cache event]
  dTLB-prefetch-misses                             [Hardware cache event]
```

continued:

```
 iTLB-loads                                         [Hardware cache event]
iTLB-load-misses                                    [Hardware cache event]
branch-loads                                        [Hardware cache event]
branch-load-misses                                  [Hardware cache event]

rNNN (see 'perf list --help' on how to encode it)  [Raw hardware event descriptor]

mem:<addr>[:access]                                 [Hardware breakpoint]

sunrpc:rpc_call_status                              [Tracepoint event]
sunrpc:rpc_bind_status                              [Tracepoint event]
sunrpc:rpc_connect_status                           [Tracepoint event]
...
sched:sched_wakeup_new                              [Tracepoint event]
sched:sched_switch                                  [Tracepoint event]
sched:sched_migrate_task                            [Tracepoint event]
sched:sched_process_free                            [Tracepoint event]
...
irq:irq_handler_entry                               [Tracepoint event]
irq:irq_handler_exit                                [Tracepoint event]
...
```

All tracepoint events can be monitored by perf

In-depth profiling of a single application

- Commands
  - `perf record` -- generate
  - `perf report` -- analyze
  - `perf annotate` -- analyze
- Samples based on PMU counter overflow (2^64)
- They all operate on perf.data files
- Annotation requires compile with `-ggdb`
  - Otherwise you just get disassembly
- Annotation also requires rebuild w/ proper path to appropriate `objdump`
  - Recent versions have a `--objdump=` option
- Can record system-wide or one single process

Recording system-wide:

```
# perf record -a sleep 30
```

Reading the report:

```
 # perf report
no symbols found in /system/bin/mpdecision, maybe install a debug package?
Failed to open /init, continuing without symbols
no symbols found in /system/bin/mksh, maybe install a debug package?
Failed to open /sbin/adbd, continuing without symbols
...
Kernel address maps (/proc/{kallsyms,modules}) were restricted.

Check /proc/sys/kernel/kptr_restrict before running 'perf record'.

If some relocation was applied (e.g. kexec) symbols may be misresolved.

Samples in kernel modules can't be resolved as well.
...
```

Annotating:

```
$ perfhost annotate --symfs out/target/product/flo/symbols
```

```
 # Events: 15K cycles
#
# Overhead          Command                     Shared Object               ...              Symbol
# ........  ...............  ..............................  ..................................
#
    61.17%             adbd  [unknown]                       [k] 0xc07c5cd4
     5.19%             adbd  dumpsys                         [.] 0x1b8f8
     4.52%             perf  [unknown]                       [k] 0xc07c3fe0
     3.46%          swapper  [unknown]                       [k] 0xc07c5d0c
     2.71%           logcat  [unknown]                       [k] 0xc029b0d0
     2.57%      kworker/0:0  [unknown]                       [k] 0xc07c5cd4
     1.76%       mpdecision  [unknown]                       [k] 0xc029a77c
     1.53%   system_server  dumpsys                         [.] 0x3c18c
     1.39%   system_server  [unknown]                       [k] 0xc0087710
     0.63%               ls  [unknown]                       [k] 0xc0008578
     0.51%             perf  dumpsys                         [.] 0x25fc8
     0.41%  ndroid.launcher  dumpsys                         [.] _Z17dvmHeapBitmapWalkPK10HeapBitmapPFvP6ObjectPvES4_
     0.39%  d.process.media  dumpsys                         [.] 0x39c18
     0.39%   system_server  dumpsys                         [.] 0x81740
     0.37%   system_server  dumpsys                         [.] 0x5226
     0.36%           logcat  dumpsys                         [.] 0x18f4
     0.36%   system_server  dumpsys                         [.] dvmAsmInstructionStart
     0.32%               ps  [unknown]                       [k] 0xc07c7940
     0.28%             perf  dumpsys                         [.] dlfree
     0.27%  ndroid.launcher  [unknown]                       [k] 0xc07c58d4
     0.27%             perf  dumpsys                         [.] memcpy
  ...
```

Support in Android

- `perf` tools in `external/linux-tools-perf`
- Will build only if `$TARGET_BUILD_VARIANT=eng`
- Otherwise the binary won't be in the AOSP
- Works the same as on the Linux command line
- `perf.data` files are automatically stored into `/data/`
- Annotation requires copying the `perf.data` file to the host
- `external/linux-tools-perf/` already patched to use cross-dev `objdump`

4. Debugging

- kgdb/kdb
- Other kernel debugging mechanisms
- JTAG

4.1. kgdb/kdb

- Built-in kernel debugger
- Two modes of operation:
  - kdb -> live analysis / peaking
    - Console/keyboard/serial
    - Magic Sysrq
  - kgdb -> source-level debugging
    - Remote gdb debugging
    - `target remote ...`
- "x86-centric" concept
- There's only so much you can do with this
- `Documentation/DocBook/kgdb.tmpl`

Internals

- Core: `kernel/debug/debug_core.c`
- Arch-specific code: `arch/*/kernel/kgdb.c`
- gdb stub: `kernel/debug/gdbstub.c`
- kdb front-end: `kernel/debug/kdb`
- kgdb I/O driver:
  - `drivers/tty/serial/kgdboc.c`
  - `drivers/usb/early/ehci-dbgp.c`
- Test suite: `drivers/misc/kgdbts.c`

4.2 Other kernel debugging mechanisms

- Crash dumps
  - kexec new kernel to dump-capture failed kernel
  - x86- and big-iron-centric
  - `Documentation/kdump/kdump.txt`
- Oops messages
  - Kernel errors/exceptions reported to dmesg
  - `Documentation/oops-tracing.txt`
- Dynamic debug
  - Dynamically-enable in-kernel debug info
  - `Documentation/dynamic-debug-howto.txt`

4.3. JTAG

- True geeks use JTAG debuggers
- See what the SoC is *really* doing
- Several vendors out there: Lauterbach, Abatron, ...
- Typically N*$1,000, where N > 2
- Open source: Flyswater 2 (HW) + OpenOCD (SW)
- Use/operation JTAG-vendor specific
- Typical:
  - gdb-based
  - Setup file to prep processor for debug
  - Need `vmlinux` file
  - Module debugging requires relocation info

Android-Agnostic User-Space Tools

- strace
- ltrace
- LTTng UST
- apitrace
- gdb/gdbserver

5.1. strace

- Classic Unix system call tracer
- Trace system calls and signals
- Relies on `ptrace()`, `PTRACE_SYSCALL`
- Pros:
  - Detailed info
  - Very simple to use
- Cons:
  - Modifies application behavior
- Included by default on Android
- `external/strace/`
- `man strace` on Linux host

- Several modes of operation:
  - Track existing PID
  - Start and track command
  - Save output in separate file
- Recommended
  - Use the `-o` flag to provide output file
  - Read output file separate from command output
- **Beware**:
  - Rumor has it that AOSP-packaged one sometimes has the wrong syscall table

```
# strace -o data/logcat.strace logcat
...
# cat data/logcat.strace
execve("/system/bin/logcat", ["logcat"], [/* 14 vars */]) = 0
mprotect(0x4000f000, 4096, PROT_READ)   = 0
open("/dev/null", O_RDWR)               = 3
fcntl64(0, F_GETFL)                     = 0x2 (flags O_RDWR)
fcntl64(1, F_GETFL)                     = 0x2 (flags O_RDWR)
fcntl64(2, F_GETFL)                     = 0x2 (flags O_RDWR)
close(3)                                = 0
gettid()                                = 798
set_tls(0x40010efc, 0x40010efc, 0, 0xfffffffc, 0x40010ffc) = 0
sigaction(SIGILL, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
sigaction(SIGABRT, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
sigaction(SIGBUS, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
sigaction(SIGFPE, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
sigaction(SIGSEGV, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
sigaction(SIGSTKFLT, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
sigaction(SIGPIPE, {0x40000a41, [], SA_RESTART|SA_SIGINFO}, NULL, 0x2a04b038) = 0
mmap2(NULL, 4092, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, 0, 0) = 0x40012000
open("/vendor/lib/liblog.so", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/system/lib/liblog.so", O_RDONLY) = 3
...
```

5.2. ltrace

- Problem:
  - Sometimes need to track library calls (.so)
  - strace can only trace system calls
- Solution:
  - ltrace
- Same concept as strace but for libraries
- Inserts breakpoints at symbol entry points
- Pros:
  - Detailed info
  - Very simple to use
- Cons:
  - Modifies application behavior
- Now included by default in Android

Resources

- `man ltrace`
- **http://ltrace.org/** (link to git repo)
- Android port in external/ltrace/
- **http://www.opersys.com/blog/ltrace-internals-140120**

5.3. LTTng UST

- User-space tracing mechanism for LTTng
- Provides integration with LTTng toolsuite
- Relies on SHM, which isn't exposed by Bionic
- **http://lttng.org/ust**
- Some Android efforts but no official support (yet)

5.4. apitrace

- "Generic" OpenGL tracing tool
- Relies on `LD_PRELOAD`
- No breakpoints = no behavior modification
- Records all calls to OpenGL
- Allows replay
- Has graphic analysis tools
- Some Android support upstream
- **http://apitrace.github.io/**
- **https://github.com/apitrace/apitrace**

5.5. gdb/gdbserver

- Classic user-space symbolic debugging
- For better or worse:
  - It's practically the only thing out there
- Relies on `ptrace()` for all operations
- Tightly integrated with GNU toolchain
- Use `-g` or `-ggdb` flags when building
- Several front-ends: DDD, Eclipse CDT, ...
- `man gdb`
- gdb also has online help in its shell
- `gdbserver` binary part of AOSP
- Everything works the same on Android

5.5.1. Target side

AOSP already takes care of debug:

- "-g" flag added to all native binaries
- Unstripped binaries in out/target/product/[PRODUCT_NAME]/symbols/...

Two ways to operate gdbserver:

- Start app for debugging with gdbserver prepended

```
# gdbserver localhost:2345 service list
```

- Attaching to running process

```
# gdbserver --attach locahost:2345 30
```

Either way, need to forward the port on the host:

```
$ adb forward tcp:2345 tcp:2345
```

5.5.2. Host side

Load file **FIRST** and then attach on host side

```
 $ arm-linux-androideabi-gdb
...
(gdb) file out/target/product/generic/symbols/system/bin/service
(gdb) target remote localhost:2345
(gdb) b main
Cannot access memory at address 0x0
Breakpoint 1 at 0x2a00146c: file frameworks/native/cmds/service/service.cpp, line 59.
(gdb) cont
Continuing.
warning: Could not load shared library symbols for 11 libraries, e.g. /system/bin/linker.
...
```

```
 Breakpoint 1, main (argc=2, argv=0xbe882b74) at frameworks/native/cmds/service/service.cpp:59
59    {
(gdb) n
60        sp<IServiceManager> sm = defaultServiceManager();
(gdb) n
59    {
(gdb) n
60        sp<IServiceManager> sm = defaultServiceManager();
(gdb) n
61        fflush(stdout);
...
```

Can also attach to system services' JNI code -- attach **FIRST**

```
$ arm-linux-androideabi-gdb
...
(gdb) target remote localhost:2345
(gdb) file out/target/product/msm8960/symbols/system/bin/app_process
(gdb) set solib-absolute-prefix out/target/product/msm8960/symbols/
(gdb) set solib-search-path out/target/product/msm8960/symbols/system/lib/
(gdb) b com_android_server_OpersysService.cpp:70
(gdb) cont
Continuing.
-----------------------------
root@android:/ # service call opersys 2 s16 adfasd
-----------------------------
[New Thread 576]
[Switching to Thread 576]

Breakpoint 1, write_native (env=0x5c94ad40, clazz=,
    ptr=, buffer=0xa4f00005)
    at frameworks/base/services/jni/com_android_server_OpersysService.cpp:72
72        if (dev == NULL) {
(gdb)
```

5.5.3. Using 'gdbclient'

```
$ pid system_server
1653
$ gdbclient app_process32 :5039 1653
[1] 13531
Attached; pid = 1653
Listening on port 5039
GNU gdb (GDB) 7.6
...
Reading symbols from /media/6921e18a-5b32-4fe8-97cc-62a85a6f351f/home/karim/opersys-dev/android/aosp-5.0.2_r1-pristine/out/target/product/flo/symbols
Remote debugging from host 127.0.0.1
warning: while parsing target library list: not well-formed (invalid token)
warning: Could not load shared library symbols for 29 libraries, e.g. 'U.
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?
readv () at bionic/libc/arch-arm/syscalls/readv.S:12
12        neg     r0, r0
Function "art_sigsegv_fault" not defined.
Breakpoint 1 (art_sigsegv_fault) pending.
ART debugging mode is enabled.
If you are debugging a native only process, you need to
re-enable normal SIGSEGV handling using this command:
  handle SIGSEGV print stop
```

```
(gdb) b com_android_server_power_PowerManagerService.cpp:126
Breakpoint 2 at 0xb404f404: file frameworks/base/services/core/jni/com_android_server_power_PowerManagerService.cpp, line 126.
(gdb) cont
Continuing.
[New Thread 30562]
[Switching to Thread 30562]

Breakpoint 2, android::nativeSetInteractive (env=0xaf6c30f0, clazz=0x9eff29ac,
    enable=1 '\001')
    at frameworks/base/services/core/jni/com_android_server_power_PowerManagerService.cpp:126
126         if (gPowerModule) {
(gdb) n
```

Android-Specific User-Space Tools

- dumpstate / bugreport
- watchprop / getprop / setprop
- schedtop
- librank
- procmem
- procrank
- showmap
- timeinfo
- log driver / logger / logcat
- EGL trace / built-in

- tombstones
- debuggerd
- input
- ioctl
- start / stop / init "disable" flag
- notify
- run-as
- schedtest
- adb

6.1. dumpstate / bugreport

- Get a complete dump of system state
- Reads from a lot of data sources
  - logcat
  - dumpsys
  - /proc
  - etc.
- Two versions:
  - `dumpstate` requires root
  - `bugreport` doesn't require root

```
usage: dumpstate [-b soundfile] [-e soundfile] [-o file [-d] [-p] [-z]] [-s] [-q]
  -o: write to file (instead of stdout)
  -d: append date to filename (requires -o)
  -z: gzip output (requires -o)
  -p: capture screenshot to filename.png (requires -o)
  -s: write output to control socket (for init)
  -b: play sound file instead of vibrate, at beginning of job
  -e: play sound file instead of vibrate, at end of job
  -q: disable vibrate
  -B: send broadcast when finished (requires -o and -p)
```

6.2. Global properties

- init process maintains set of global properties
- Can:
  - List properties: `getprop`
  - Set properties: `setprop`
  - Watch properties: `watchprop`
- Property files:
  - `/default.prop`
  - `/system/build.prop`
  - `/system/default.prop`
  - `/data/local.prop`
- Property triggers in init's `.rc` files

6.3. schedtop

- Similar to plain top
- Cumulative execution time of processes

```
schedtop [-d <delay>] [-bitamun]
        -d refresh every <delay> seconds
        -b batch - continous prints instead of refresh
        -i hide idle tasks
        -t show threads
        -a use alternate screen
        -m use millisecond precision
        -u use microsecond precision
        -n use nanosecond precision
```

```
Processes: 67, Threads 412
  TID    -------- SINCE LAST --------       ------------- TOTAL -------------
  PID     EXEC_TIME   DELAY_TIME SCHED       EXEC_TIME      DELAY_TIME   SCHED NAME
    1   0.000000000  0.000000000     0     2.280000000     0.630000000     248 /init
    2   0.000000000  0.000000000     0     0.020000000     0.010000000      45 kthreadd
    3   0.000000000  0.000000000     0     0.040000000     0.030000000      35 ksoftirqd/0
    5   0.000000000  0.000000000     0     0.000000000     0.010000000       4 kworker/u:0
    6   0.000000000  0.000000000     0     0.000000000     0.000000000       2 khelper
    7   0.000000000  0.000000000     1     0.010000000     0.010000000      83 sync_supers
    8   0.000000000  0.000000000     0     0.000000000     0.000000000       7 bdi-default
    9   0.000000000  0.000000000     0     0.000000000     0.000000000       2 kblockd
   10   0.000000000  0.000000000     0     0.000000000     0.000000000       2 rpciod
   11   0.010000000  0.000000000    96     0.280000000     1.220000000    6700 kworker/0:1
   12   0.000000000  0.000000000     0     0.000000000     0.000000000       3 kswapd0
...
```

6.4. librank

Print library memory usage

```
 Usage: librank [ -P | -L ] [ -v | -r | -p | -u | -h ]

Sort options:
    -v  Sort processes by VSS.
    -r  Sort processes by RSS.
    -p  Sort processes by PSS.
    -u  Sort processes by USS.
        (Default sort order is PSS.)
    -P /path  Limit libraries displayed to those in path.
    -R  Reverse sort order (default is descending).
    -h  Display this help screen.
```

```
 RSStot       VSS       RSS       PSS       USS  Name/PID
55386K                                          /dev/ashmem/dalvik-heap
          29340K    29340K    23506K    23272K    com.android.systemui [645]
          13680K    13680K     7753K     7516K    com.android.launcher [765]
          11240K    11240K     5406K     5172K    system_server [565]
           7664K     7664K     1628K     1384K    com.android.phone [737]
           7552K     7552K     1521K     1280K    android.process.media [692]
           7392K     7392K     1326K     1076K    android.process.acore [818]
           7228K     7228K     1184K      940K    com.android.inputmethod.latin [710]
           7108K     7108K     1031K      784K    com.android.email [1091]
...

40517K                                          anon_inode:dmabuf
          39972K    39972K    25758K    11544K    /system/bin/surfaceflinger [253]
          16172K    16172K     8142K      132K    system_server [565]
          11884K    11884K     5944K        4K    com.android.launcher [765]
            964K      964K      673K      408K    com.android.systemui [645]
            408K        0K        0K        0K    /system/bin/mediaserver [256]
             40K        0K        0K        0K    /system/bin/qseecomd [341]

19489K                                          /dev/ashmem/dalvik-aux-structure
           1480K     1456K     1194K     1184K    system_server [565]
           1812K     1740K     1116K     1088K    com.android.email [1091]
           1628K     1552K     1087K     1068K    com.android.phone [737]
           1824K     1740K     1076K     1044K    com.android.contacts [904]
           1656K     1572K     1050K     1028K    android.process.media [692]
           1760K     1684K      982K      944K    com.android.settings [801]
...
```

6.5. procmem

See PID's memory usage

```
Usage: procmem [ -w | -W ] [ -p | -m ] [ -h ] pid
    -w  Displays statistics for the working set only.
    -W  Resets the working set of the process.
    -p  Sort by PSS.
    -m  Sort by mapping order (as read from /proc).
    -h  Hide maps with no RSS.
```

```
  # procmem 565
    Vss      Rss      Pss      Uss     ShCl     ShDi     PrCl     PrDi  Name
-------  -------  -------  -------  -------  -------  -------  -------
     4K       4K       4K       4K       0K       0K       4K       0K  ??
     4K       4K       0K       0K       4K       0K       0K       0K  /dev/ashmem/SurfaceFlinger
     4K       4K       2K       0K       4K       0K       0K       0K  /system/app/SettingsProvider.apk
     4K       4K       2K       0K       4K       0K       0K       0K  /system/app/SettingsProvider.apk
     0K       0K       0K       0K       0K       0K       0K       0K  /system/framework/framework_ext.jar
     8K       8K       0K       0K       8K       0K       0K       0K  /system/lib/libstagefright_yuv.so
     4K       4K       0K       0K       4K       0K       0K       0K  /system/lib/libstagefright_yuv.so
     4K       4K       4K       4K       0K       0K       4K       0K  /system/lib/libstagefright_yuv.so
     8K       8K       8K       8K       0K       0K       8K       0K  /system/app/SettingsProvider.apk
     0K       0K       0K       0K       0K       0K       0K       0K  /system/framework/core-junit.jar
     8K       8K       8K       8K       0K       0K       8K       0K  /system/framework/core-junit.jar
    32K      32K       8K       8K      24K       0K       8K       0K  /system/framework/core-junit.jar
     8K       8K       8K       8K       0K       0K       8K       0K  /dev/ashmem/dalvik-aux-structure
    64K      64K       7K       0K      64K       0K       0K       0K  /system/lib/libm.so
     0K       0K       0K       0K       0K       0K       0K       0K  /system/lib/libm.so
     4K       4K       0K       0K       4K       0K       0K       0K  /system/lib/libm.so
    16K      16K       8K       8K       8K       0K       8K       0K  /system/lib/libm.so
     0K       0K       0K       0K       0K       0K       0K       0K  /system/framework/android.policy.jar
     4K       4K       0K       0K       4K       0K       0K       0K  /system/framework/android.policy.jar
     8K       8K       8K       8K       0K       0K       8K       0K  /system/framework/android.policy.jar
...
```

6.6. procrank

See processes' memory usage, in order

```
Usage: procrank [ -W ] [ -v | -r | -p | -u | -h ]
    -v  Sort by VSS.
    -r  Sort by RSS.
    -p  Sort by PSS.
    -u  Sort by USS.
        (Default sort order is PSS.)
    -R  Reverse sort order (default is descending).
    -w  Display statistics for working set only.
    -W  Reset working set of all processes.
    -h  Display this help screen.
```

```
    # procrank
      PID      Vss      Rss      Pss      Uss  cmdline
      565   77364K   77216K   36443K   24816K  system_server
      645   63492K   63172K   31496K   28356K  com.android.systemui
      253   64300K   51900K   31349K   15944K  /system/bin/surfaceflinger
      765   67408K   67116K   28784K   19532K  com.android.launcher
      818   35496K   35392K    7159K    5356K  android.process.acore
      737   35084K   34984K    6936K    5444K  com.android.phone
      254   37100K   36908K    6758K    4392K  zygote
      710   34420K   34340K    6347K    4916K  com.android.inputmethod.latin
      692   33404K   33236K    5879K    4644K  android.process.media
     1091   32892K   32736K    5436K    4232K  com.android.email
      256    9392K    8980K    5018K    4812K  /system/bin/mediaserver
      904   31524K   31356K    4505K    3336K  com.android.contacts
     1141   31468K   31316K    4336K    3160K  com.android.mms
     1052   31676K   31508K    4252K    3064K  com.android.providers.calendar
      801   31016K   30916K    4190K    2988K  com.android.settings
     1230   30896K   30728K    3955K    2784K  com.android.calendar
...
```

6.7. showmap

See objects mapped to process' address space

```
showmap [-t] [-v] [-c] <pid>
       -t = terse (show only items with private pages)
       -v = verbose (don't coalesce maps with the same name)
       -a = addresses (show virtual memory map)
```

```
# showmap 565
 virtual                     shared   shared  private  private
    size      RSS      PSS    clean    dirty    clean    dirty    # object
-------- -------- -------- -------- -------- -------- -------- ---- ------------------------------
      68       60       60        0        0       60        0    1 /data/dalvik-cache/system@app@SettingsProvider.apk@classes.dex
     336      276      135      192        0       84        0    1 /data/dalvik-cache/system@framework@android.policy.jar@classes.dex
    1348       32        1       32        0        0        0    1 /data/dalvik-cache/system@framework@apache-xml.jar@classes.dex
     960       92        6       92        0        0        0    1 /data/dalvik-cache/system@framework@bouncycastle.jar@classes.dex
     124      112      112        0        0      112        0    1 /data/dalvik-cache/system@framework@com.quicinc.cne.jar@classes.dex
      28       12        0       12        0        0        0    1 /data/dalvik-cache/system@framework@core-junit.jar@classes.dex
    3320     1848      278     1780        0       68        0    1 /data/dalvik-cache/system@framework@core.jar@classes.dex
    1468       88        8       88        0        0        0    1 /data/dalvik-cache/system@framework@ext.jar@classes.dex
   11156     6216     1307     5680        0      536        0    1 /data/dalvik-cache/system@framework@framework.jar@classes.dex
     776       36        1       36        0        0        0    1 /data/dalvik-cache/system@framework@framework_ext.jar@classes.dex
    2384     1860     1593      440        0     1420        0    1 /data/dalvik-cache/system@framework@services.jar@classes.dex
      32       32       32        0        0       32        0    1 /data/data/com.android.providers.settings/databases/settings.db-shm
      32       32       32        0        0       32        0    1 /data/system/locksettings.db-shm
      48       32        0        0       32        0        0    1 /dev/__properties__ (deleted)
    8192       16       16        0        0        0       16    4 /dev/ashmem/CursorWindow: /data/data/com.android.providers.settings/databases
       4        4        0        4        0        0        0    1 /dev/ashmem/SurfaceFlinger read-only heap (deleted)
  ...
```

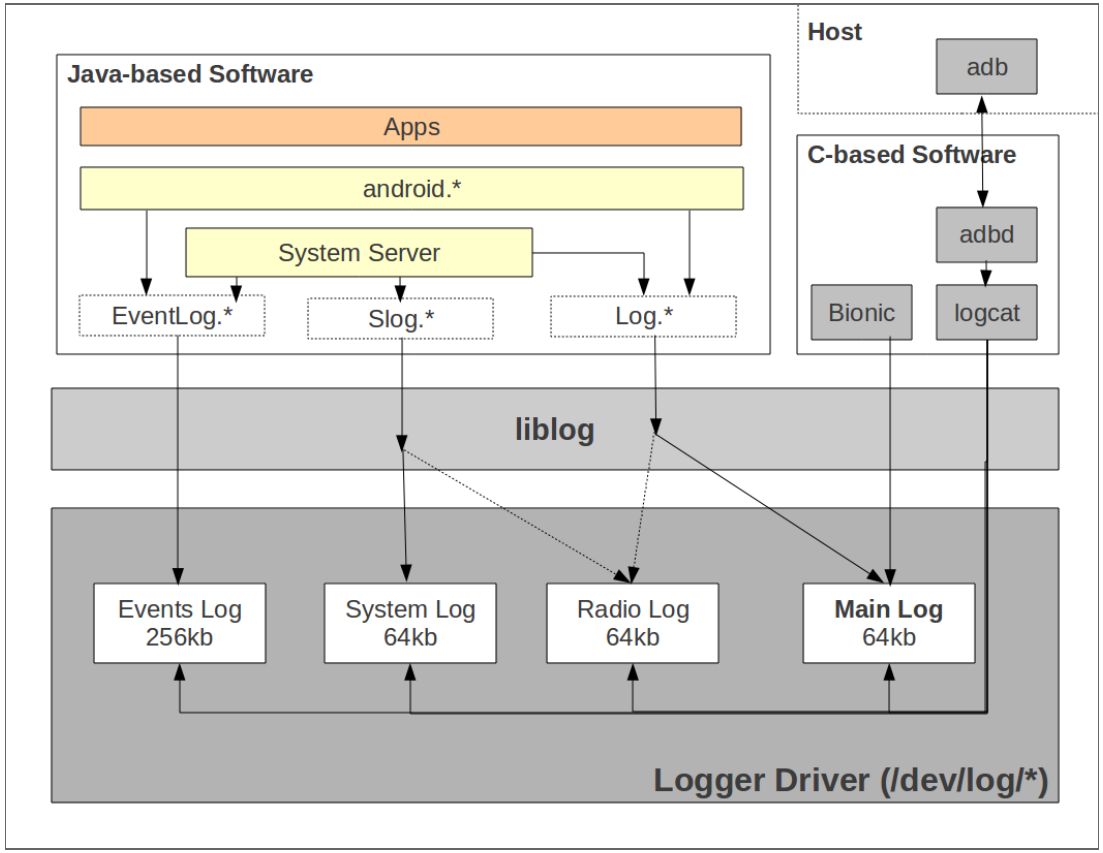
6.8. timeinfo

Report:

- realtime
- uptime
- awake percentage
- sleep percentage

```
# timeinfo
986408 986416 100 0
```

6.9. Logger / logcat



```
 Usage: logcat [options] [filterspecs]
options include:
  -s              Set default filter to silent.
                  Like specifying filterspec '*:s'
  -f <filename>   Log to file. Default to stdout
  -r [<kbytes>]   Rotate log every kbytes. (16 if unspecified). Requires -f
  -n <count>      Sets max number of rotated logs to <count>, default 4
  -v <format>     Sets the log print format, where <format> is one of:

                  brief process tag thread raw time threadtime long

  -c              clear (flush) the entire log and exit
  -d              dump the log and then exit (don't block)
  -t <count>      print only the most recent <count> lines (implies -d)
  -g              get the size of the log's ring buffer and exit
  -b <buffer>     Request alternate ring buffer, 'main', 'system', 'radio'
                  or 'events'. Multiple -b parameters are allowed and the
                  results are interleaved. The default is -b main -b system.
  -B              output the log in binary
  ...
```

```
 --------- beginning of /dev/log/main
I/BOOT    (  150): MSM target 'msm8960', SoC 'Surf', HwID '109', SoC ver '65536'
I/qcom-bluetooth(  289): /system/etc/init.qcom.bt.sh: init.qcom.bt.sh config = onboot
I/qrngd   (  275): qrngd has started:
I/qrngd   (  275): Reading device:'/dev/hw_random' updating entropy for device:'/dev/random'
I/DMM     (  305): DMM available. movable_start_bytes at
I/DEBUG   (  251): debuggerd: Jan 10 2014 20:38:46
D/PPDaemon(  287): isHDMIPrimary: HDMI is not primary display
D/PPDaemon(  287): CABL version 1.0.20120512
I/qcom-bluetooth(  311): /system/etc/init.qcom.bt.sh: Bluetooth Address programmed successfully
--------- beginning of /dev/log/system
I/Vold    (  246): Vold 2.1 (the revenge) firing up
E/PPDaemon(  287): Failed to open the config file!
D/Vold    (  246): Volume sdcard state changing -1 (Initializing) -> 0 (No-Media)
D/QSEECOMD: (  293): qseecom listener services process entry PPID = 1
D/QSEECOMD: (  293): Parent qseecom daemon process paused!!
D/QSEECOMD: (  341): QSEECOM DAEMON RUNNING
D/QSEECOMD: (  341): qseecom listener service threads starting!!!
D/QSEECOMD: (  341): Total listener services to start = 2
D/QSEECOMD: (  341): Init dlopen(libdrmtime.so, RTLD_NOW) succeeds
D/QSEECOMD: (  341): Init::Init dlsym(g_FSHandle atime_start) succeeds
...
```

```
 # log
USAGE: log [-p priorityChar] [-t tag] message
    priorityChar should be one of:
        v,d,i,w,e
```

6.10. EGL trace / built-in

For tracing the GL calls

http://groleo.wordpress.com/2013/03/16/android-opengl-es-tracer/

https://developer.android.com/tools/help/gltracer.html

6.11. tombstones

Closest thing to "core dumps"

```
 # ls /data/tombstones/ -al
drwxrwx--x system   system            1970-01-01 06:51 dsps
drwxrwx--x system   system            1970-01-01 06:51 lpass
drwxrwx--x system   system            1970-01-01 06:51 mdm
drwxrwx--x system   system            1970-01-01 06:51 modem
drwxrwx--x system   system            1970-01-01 06:51 wcnss
```

Usually actual files are called `tombstone_XX` where XX is a number.

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
Build fingerprint: 'Android/aosp_arm/generic:4.4/KRT16M/eng.karim.20131112.142320:eng/test-keys'
Revision: '0'
pid: 1150, tid: 1150, name: vdc  >>> vdc <<<
signal 13 (SIGPIPE), code -6 (SI_TKILL), fault addr --------
    r0 ffffffe0  r1 b7a5c028  r2 00000457  r3 00000888
    r4 b6ef01a4  r5 b7a5c028  r6 00000457  r7 00000004
    r8 00001000  r9 00000000  sl b6f00ee4  fp 0000000c
    ip b6efe2fc  sp bed41a30  lr b6ecb89f  pc b6ec7178  cpsr 20000010
    d0  a9c01b6937fe9a6b  d1  0000000000000000
    d2  0000000000000000  d3  0000000000000000
    d4  0000000000000000  d5  41cbff4d35800000
    d6  3f50624dd2f1a9fc  d7  c1d58ff925dc7ae1
    d8  0000000000000000  d9  0000000000000000
    d10 0000000000000000  d11 0000000000000000
    d12 0000000000000000  d13 0000000000000000
    d14 0000000000000000  d15 0000000000000000
    scr 00000010
...
```

```
   backtrace:
      #00  pc 00020178  /system/lib/libc.so (write+12)
      #01  pc 0002489d  /system/lib/libc.so (__sflush+54)
      #02  pc 00014393  /system/lib/libc.so (fclose+54)
      #03  pc 0000d939  /system/lib/libc.so
      #04  pc 0000eecc  /system/lib/libc.so (pthread_once+104)
      #05  pc 0000db93  /system/lib/libc.so
      #06  pc 00027ded  /system/lib/libc.so (__cxa_finalize+156)
      #07  pc 00027fe5  /system/lib/libc.so (exit+6)
      #08  pc 00000b03  /system/bin/vdc
      #09  pc 0000e23b  /system/lib/libc.so (__libc_init+50)
      #10  pc 000007f0  /system/bin/vdc

   stack:
            bed419f0  00000000
            bed419f4  00000000
            bed419f8  00000000
         ...
```

6.12. debuggerd

- Daemon running in background
- Allows catching crashing processes
- Set `debug.db.uid` to "greater than" UID to trigger
- Linker has code for latching to `debuggerd`

```
I/DEBUG   (  365): *********************************************************
I/DEBUG   (  365): * Process 984 has been suspended while crashing.  To
I/DEBUG   (  365): * attach gdbserver for a gdb connection on port 5039:
I/DEBUG   (  365): *
I/DEBUG   (  365): *     adb shell gdbserver :5039 --attach 984 &
I/DEBUG   (  365): *
I/DEBUG   (  365): * Press HOME key to let the process continue crashing.
I/DEBUG   (  365): *********************************************************
```

6.13. input

Send input to input layer

```
 # input
usage: input ...
       input text <string>
       input keyevent <key code number or name>
       input tap <x> <y>
       input swipe <x1> <y1> <x2> <y2>
```

6.14. ioctl

Send `ioctl()` calls to device driver

```
ioctl [-l <length>] [-a <argsize>] [-rdh] <device> <ioctlnr>
 -l <lenght>   Length of io buffer
 -a <argsize>  Size of each argument (1-8)
 -r            Open device in read only mode
 -d            Direct argument (no iobuffer)
 -h            Print help
```

6.15. Control `init` services

- Stop service: `stop servicename`
- Start service: `start servicename`
- If no service name is specified:
  - zygote
  - surfaceflinger
- Can mark service as `disabled` in `.rc` files

6.16. notify

Monitor path using inotify kernel functionality (`man inotify`)

```
Usage: notify [-m eventmask] [-c count] [-p] [-v verbosity] path [path ...]
```

6.17. run-as

Run a command under a given package's user ID

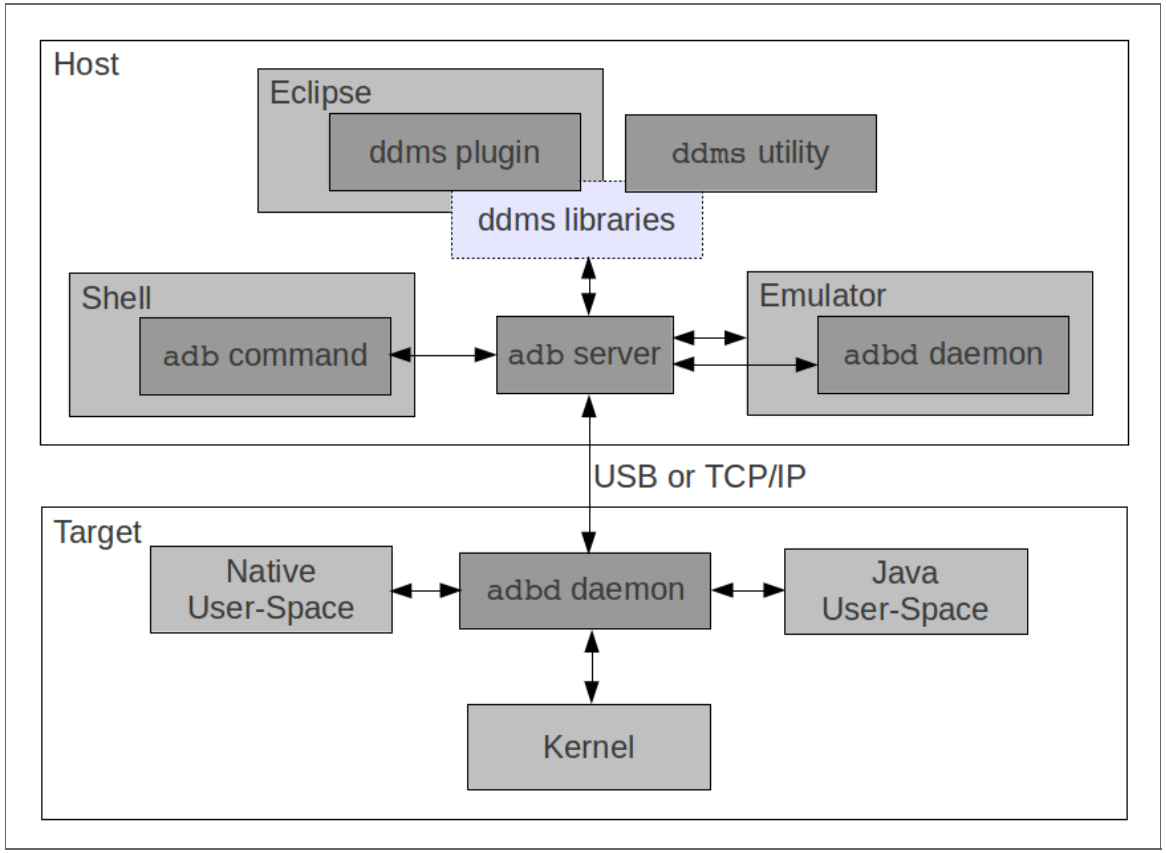
```
Usage: run-as <package-name> <command> [<args>]
```

6.18. schedtest

Test scheduler's ability to wake processes up after 1ms

```
 # schedtest
max 3449    avg 1171
max 3418    avg 1170
max 3205    avg 1167
max 2380    avg 1162
max 3449    avg 1169
max 9340    avg 1179
max 3418    avg 1168
max 3388    avg 1168
max 3418    avg 1170
max 3388    avg 1168
max 3418    avg 1167
...
```

6.19. adb


Host
Eclipse
ddms plugin
ddms utility
ddms libraries
Shell
adb command
adb server
Emulator
adbd daemon
USB or TCP/IP
Target
Native
User-Space
adbd daemon
Java
User-Space
Kernel

Java Tools

- dalvikvm
- dvz
- app_process
- ddms
- dexdump
- jdb/jdwp
- Android Studio integration
- junit
- traceview / dmtracedump
- Memory usage analysis

7.1. dalvikvm

- Raw Dalvik VM
- Can't run Android code
- Seldom used

```
 # dalvikvm -help

dalvikvm: [options] class [argument ...]
dalvikvm: [options] -jar file.jar [argument ...]

The following standard options are recognized:
  -classpath classpath
  -Dproperty=value
  -verbose:tag  ('gc', 'jni', or 'class')
  -ea[:<package name>... |:<class name>]
  -da[:<package name>... |:<class name>]
   (-enableassertions, -disableassertions)
  -esa
  -dsa
   (-enablesystemassertions, -disablesystemassertions)
  -showversion
  -help
  ...
```

7.2. dvz

Requests Zygote to start a specific class

```
 # <userinput>dvz --help</userinput>
Usage: dvz [--help] [-classpath <classpath>]
    [additional zygote args] fully.qualified.java.ClassName [args]

Requests a new Dalvik VM instance to be spawned from the zygote
process. stdin, stdout, and stderr are hooked up. This process remains
while the spawned VM instance is alive and forwards some signals.
The exit code of the spawned VM instance is dropped.
```

- Not built by default
- Seldom used

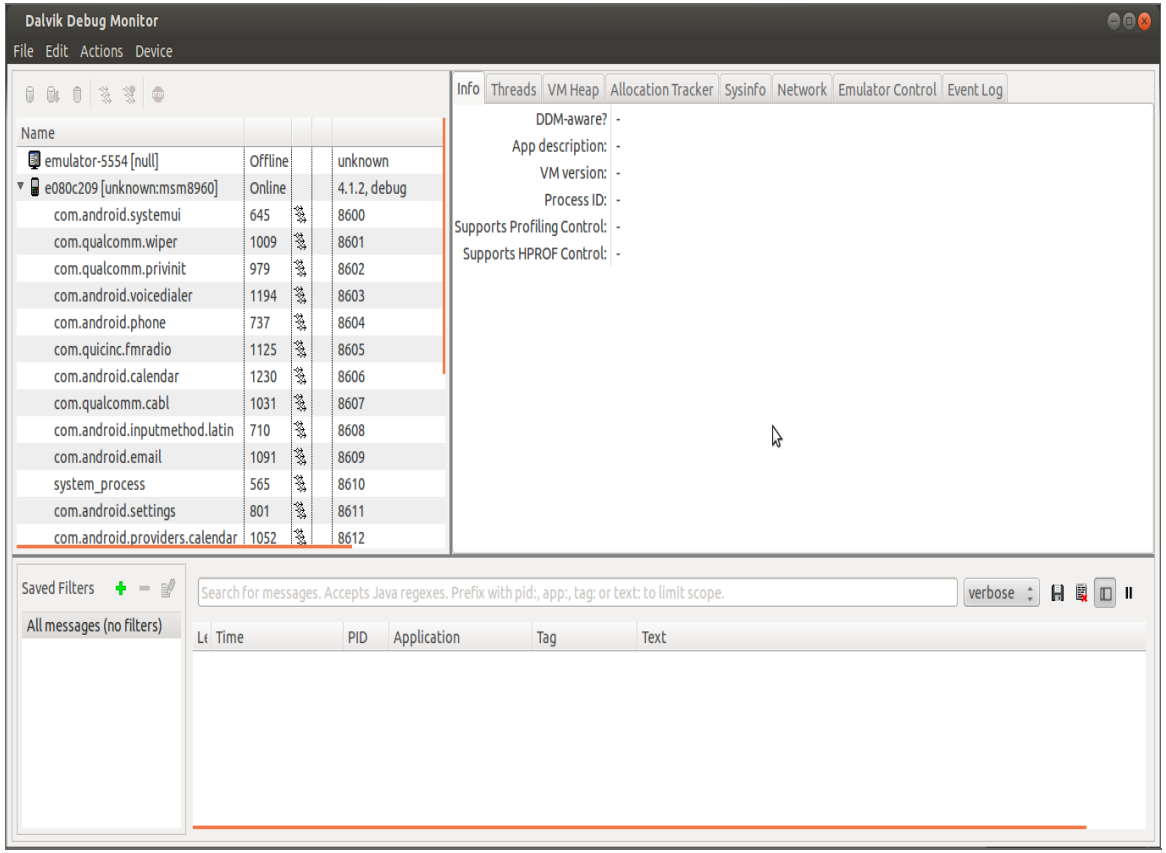
7.3. app_process

- *Magic* command to start Dalvik VM with Android Runtime
- Entirely coded in C
- Used to start initial Zygote in init.rc

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server
```

- Also used to start a variety of Java-coded framework commands:
  - am
  - pm
  - wm
  - svc
  - monkey
  - ...

7.4. ddms


Dalvik Debug Monitor
File Edit Actions Device
Name
emulator-5554 [null] Offline unknown
e080c209 [unknown:msm8960] Online 4.1.2, debug
com.android.systemui 645 8600
com.qualcomm.wiper 1009 8601
com.qualcomm.privinit 979 8602
com.android.voicedialer 1194 8603
com.android.phone 737 8604
com.quicinc.fmradio 1125 8605
com.android.calendar 1230 8606
com.qualcomm.cabl 1031 8607
com.android.inputmethod.latin 710 8608
com.android.email 1091 8609
system_process 565 8610
com.android.settings 801 8611
com.android.providers.calendar 1052 8612
Info Threads VM Heap Allocation Tracker Sysinfo Network Emulator Control Event Log
DDM-aware? -
App description: -
VM version: -
Process ID: -
Supports Profiling Control: -
Supports HPROF Control: -
Saved Filters
All messages (no filters)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose
Time PID Application Tag Text

Dalvik Debug Monitor Service

- Starting Dalvik instances register with adb
- `adb jdwp` lists all such-registered processes
- ddms connects over adb to Dalvik processes
- Provides host-side port number allowing host-side debugger to connect to remote Dalvik process
- Can only debug Dalvik instances that start *after* adb is started:
  - Important if adb isn't started at boot time
- Recently being deprecated in favor of `monitor`
- Unfortunately, monitor doesn't build by default in AOSP

7.5. dexdump

Dex file disassembler

```
dexdump: [-c] [-d] [-f] [-h] [-i] [-l layout] [-m] [-t tempfile] dexfile...

-c : verify checksum and exit
-d : disassemble code sections
-f : display summary information from file header
-h : display file header details
-i : ignore checksum failures
-l : output layout, either 'plain' or 'xml'
-m : dump register maps (and nothing else)
-t : temp file name (defaults to /sdcard/dex-temp-*)
```

```
 # dexdump /system/app/Launcher2.apk
Processing '/system/app/Launcher2.apk'...
Opened '/system/app/Launcher2.apk', DEX version '035'
Class #0            -
  Class descriptor  : 'Landroid/support/v13/app/FragmentCompat$FragmentCompatImpl;'
  Access flags      : 0x0600 (INTERFACE ABSTRACT)
  Superclass        : 'Ljava/lang/Object;'
  Interfaces        -
  Static fields     -
  Instance fields   -
  Direct methods    -
  Virtual methods   -
    #0              : (in Landroid/support/v13/app/FragmentCompat$FragmentCompatImpl;)
      name          : 'setMenuVisibility'
      type          : '(Landroid/app/Fragment;Z)V'
      access        : 0x0401 (PUBLIC ABSTRACT)
      code          : (none)
      ...
```

7.6. jdb/jdwp

- jdb = Java's gdb
- jdwp = Java Debug Wire Protocol
- In principle, can use jdb to debug Java processes
- In practice: use Android Studio
- References:
  http://docs.oracle.com/javase/1.5.0/docs/guide/jpda/jdwp-spec.html
  http://docs.oracle.com/javase/7/docs/technotes/guides
  /jpda/jdwpTransport.html

7.7. Android Studio integration

- ddms/Studio integration
- Starting debug with Eclipse
- Debugging
- Debugging multiple processes

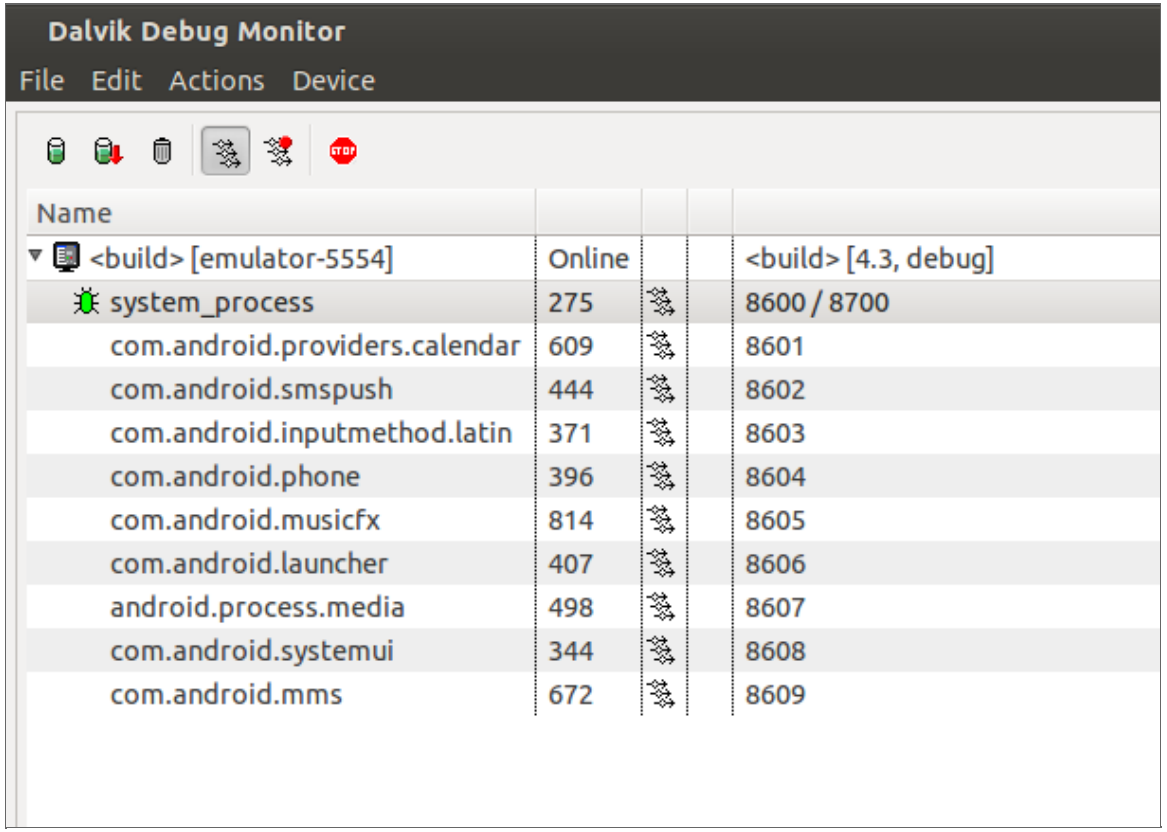
7.7.1. ddms/Studio integration

- Make sure you don't have libgail18 installed

```
sudo dpkg --force-depends -r libgail18
```

- Start Studio
- Start Monitor ("Android" icon on toolbar)
- Each process has a separate host-side socket
- Select the process you want to debug:
  - It'll get port 8700

- Go back to Studio:
  - Run->Edit Configurations->"+"
  - Remote->Port: 8700
- Apply & Debug
- Go back to DDMS:
  - Check that the little green bug is beside your process in ddms
- You're now ready to debug


Dalvik Debug Monitor
File Edit Actions Device
Name
<build> [emulator-5554] Online <build> [4.3, debug]
system_process 275 8600 / 8700
com.android.providers.calendar 609 8601
com.android.smspush 444 8602
com.android.inputmethod.latin 371 8603
com.android.phone 396 8604
com.android.musicfx 814 8605
com.android.launcher 407 8606
android.process.media 498 8607
com.android.systemui 344 8608
com.android.mms 672 8609

7.7.2. Multiple processes

- Select process in Monitor
- Go back to Studio and start a new debugging session
- Each process will now have a green bug beside it

7.8. junit

- Java's unit testing framework
- Used extensively in Android
- References:
  https://developer.android.com/tools/testing/testing_android.html
  http://www.vogella.com/tutorials/JUnit/article.html
  http://junit.org/

7.9. traceview / dmtracedump

- In-app instrumentation
- Tools to view traces;
  - traceview
  - dmtracedump
- Reference
  https://developer.android.com/tools/debugging/debugging-tracing.html

7.10. Memory usage analysis

- Two tools:
  - ddms
  - Eclipse Memory Analyzer (MAT)
- References:
  http://android-developers.blogspot.com/2011/03/memory-analysis-for-android.html
  http://www.vogella.com/tutorials/EclipseMemoryAnalyzer/article.html
  http://www.eclipse.org/mat/

System Services Interfacing

- dumpsys
- service (espc. “service call” and aidl files)
- am
- pm
- svc
- monkey
- ANR dumps

8.1. dumpsys

- Allows you to poke system services
- Calls the system service's `dump()` function
- By default will dump all system services

```
# dumpsys
```

- Can dump just one system service

```
# dumpsys statusbar
```

- C-based tool

8.2. service

- Interact with system services

```
 Usage: service [-h|-?]
        service list
        service check SERVICE
        service call SERVICE CODE [i32 INT | s16 STR] ...
Options:
   i32: Write the integer INT into the send parcel.
   s16: Write the UTF-16 string STR into the send parcel.
```

- See system service's aidl file to get "CODE" and parameter list
- C-based tool

8.3. am

- Interact with the activity manager
- Allows you to send intents on the command line (very powerful)

```
 # am
usage: am [subcommand] [options]
usage: am start [-D] [-W] [-P <FILE>] [--start-profiler <FILE>]
               [--R COUNT] [-S] [--opengl-trace] <INTENT>
       am startservice <INTENT>
       am force-stop <PACKAGE>
       am kill <PACKAGE>
       am kill-all
       am broadcast <INTENT>
       am instrument [-r] [-e <NAME> <VALUE>] [-p <FILE>] [-w]
               [--no-window-animation] <COMPONENT>
       am profile start <PROCESS> <FILE>
       am profile stop [<PROCESS>]
       am dumpheap [flags] <PROCESS> <FILE>
       ...
```

8.4. pm

Interact with package manager

```
usage: pm list packages [-f] [-d] [-e] [-s] [-3] [-i] [-u] [FILTER]
       pm list permission-groups
       pm list permissions [-g] [-f] [-d] [-u] [GROUP]
       pm list instrumentation [-f] [TARGET-PACKAGE]
       pm list features
       pm list libraries
       pm path PACKAGE
       pm install [-l] [-r] [-t] [-i INSTALLER_PACKAGE_NAME] [-s] [-f]
                  [--algo <algorithm name> --key <key-in-hex> --iv <IV-in-hex>] PATH
       pm uninstall [-k] PACKAGE
       pm clear PACKAGE
       pm enable PACKAGE_OR_COMPONENT
```

8.5. svc

Interact with various system services

```
Available commands:
   help     Show information about the subcommands
   power    Control the power manager
   data     Control mobile data connectivity
   wifi     Control the Wi-Fi manager
   usb      Control Usb state
```

8.6. monkey

- Interact with UI
- Can take scripts

```
usage: monkey [-p ALLOWED_PACKAGE [-p ALLOWED_PACKAGE] ...]
              [-c MAIN_CATEGORY [-c MAIN_CATEGORY] ...]
              [--ignore-crashes] [--ignore-timeouts]
              [--ignore-security-exceptions]
              [--monitor-native-crashes] [--ignore-native-crashes]
              [--kill-process-after-error] [--hprof]
              [--pct-touch PERCENT] [--pct-motion PERCENT]
              [--pct-trackball PERCENT] [--pct-syskeys PERCENT]
              [--pct-nav PERCENT] [--pct-majornav PERCENT]
              [--pct-appswitch PERCENT] [--pct-flip PERCENT]
              [--pct-anyevent PERCENT] [--pct-pinchzoom PERCENT]
              [--pkg-blacklist-file PACKAGE_BLACKLIST_FILE]
```

8.7. ANR dumps

- If an app hangs, it'll generate an "Application Not Responding" event
- Info about those ANRs is dumped in files in `/data/anr`

Other Tools and Techniques

- Power management / DVFS
  - Documentation/cpu-freq/*
  - **http://processors.wiki.ti.com/index.php/DVFS_User_Guide**
- DS-5
  http://www.arm.com/products/tools/software-tools/ds-5/index.php
- sqlite3
- Using screen overlays (a-la CPU perf by Status Bar)

glibc User-Space

- Running "standard" glibc-based code with Android
- Integrating glibc in Android filesystem
- Building glibc-linked code to run with Android
- Interfacing between a glibc-based stack and the Android-stack

See courseware at
http://www.opersys.com/training/embedded-android

Thank You!

karim.yaghmour@opersys.com