Android Internals

Lecture 3

Operating Systems Practical

19 October 2016

OSP
logcat crunch

Android Arhitecture

Linux Kernel

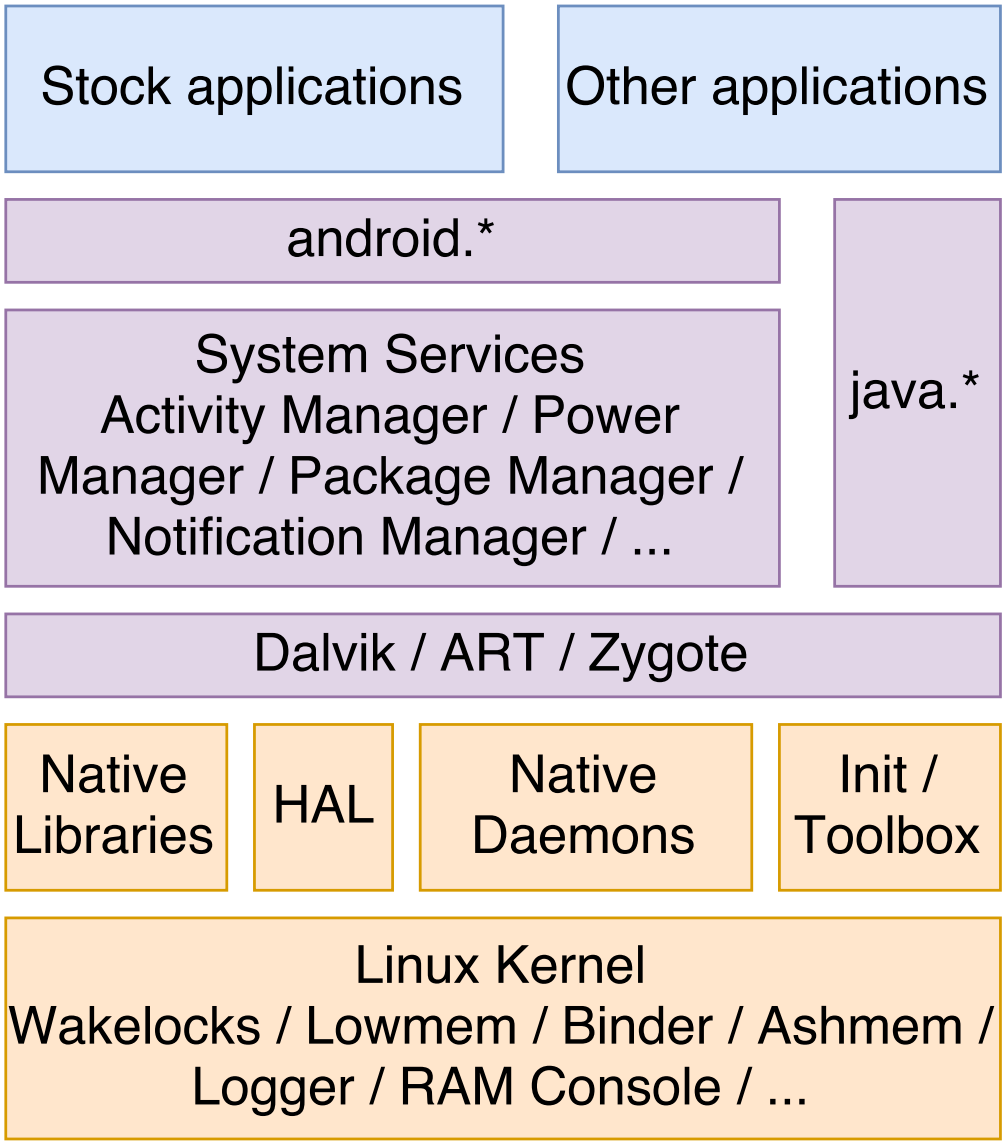
Binder

Android Framework

Managers

Outline

Android Architecture



Outline

Linux Kernel

- "Androidized" kernel
- Hundreds of patches over the standard kernel
- Device-specific functionality, fixes, enhancements
- Many features get into the mainline kernel
- "Androidisms"
  - Wakelocks
  - Low-Memory Killer
  - Binder
  - Anonymous Shared Memory
  - Alarm
  - Logger

Wakelocks

- On desktops and laptops
  - The user decides when the system goes to sleep
- The Android kernel goes to sleep as often as possible
- Sometimes you want to keep the system from going to sleep
  - Input from the user, critical operations
- Wakelocks keep the system awake

Wakelocks

- A wakelock must be obtained by the application when it needs to stay awake
  - Apps use abstractions that handle locking
  - Apps can request wakelocks directly from PowerManager Service
  - Device drivers call in-kernel wakelock primitives
- Equivalent included in mainline, from Linux 3.5
  - Autosleep
  - `epoll()` flag `EPOLLWAKEUP`

Low-Memory Killer

- Linux OOM killer
- Prevents the activation of the OOM killer (system unlikely to run out of memory)
- Kills processes with components unused for a long time
- Based on OOM adjustments mechanism
  - Different OOM kill priorities for different processes
- The userspace may control OOM killing policies
- Policies applied at startup by init
- Modified and enforced by Activity Manager

Low-Memory Killer

- Levels assigned to processes based on their components
  - Levels from -17 to 15 (high -> killed)
- Threshold (MinFree) for each type of process
  - Foreground_app - application in foreground
  - Visible_app - visible but not in foreground
  - Secondary_server - service
  - Hidden_app - hidden, needed by a running app
  - Content_provider - provide data
  - Empty_app - app not active
- Starts killing when the threshold is reached
- Included in mainline, from Linux 3.10

Anonymous Shared Memory (ashmem)

- IPC mechanism
- SysV IPC can lead to resource leakage in the kernel (vulnerability)
- File-based, reference-counted
- Similar to POSIX SHM, differences:
  - Uses reference counting to destroy the memory regions
  - Shrink mapped regions when the system needs memory
  - To shrink a region it must be unpinned

Anonymous Shared Memory (ashmem)

- First process creates region, uses Binder to share descriptor with other processes
- System services rely on ashmem, through IMemory interface
  - Surface Flinger, Audio Flinger
- Driver included in the staging tree from Linux 3.3

Alarm

- Uses the RTC and HRT functionalities
- `setitimer()`
  - Generate a signal when the time expires
  - Based on HRT
  - Does not work when the system is suspended
  - The application receives the signal when the device wakes up
- Using RTC, the alarm will be fired even if the system is suspended
  - RTC hardware device
- Uses HRT by default
- When the system is about to suspend, it uses RTC
- Apps use alarms even when the system is suspended

Alarm

- `/dev/alarm` character device, `ioctl()`
- SystemClock, AlarmManager class rely on the driver
  - SystemClock - obtain and set time
  - AlarmManager - provide alarms to apps
- The driver and AlarmManager use WakeLocks
  - The app that receives the alarm runs before the system is suspended again
- Included in mainline, from Linux 3.20

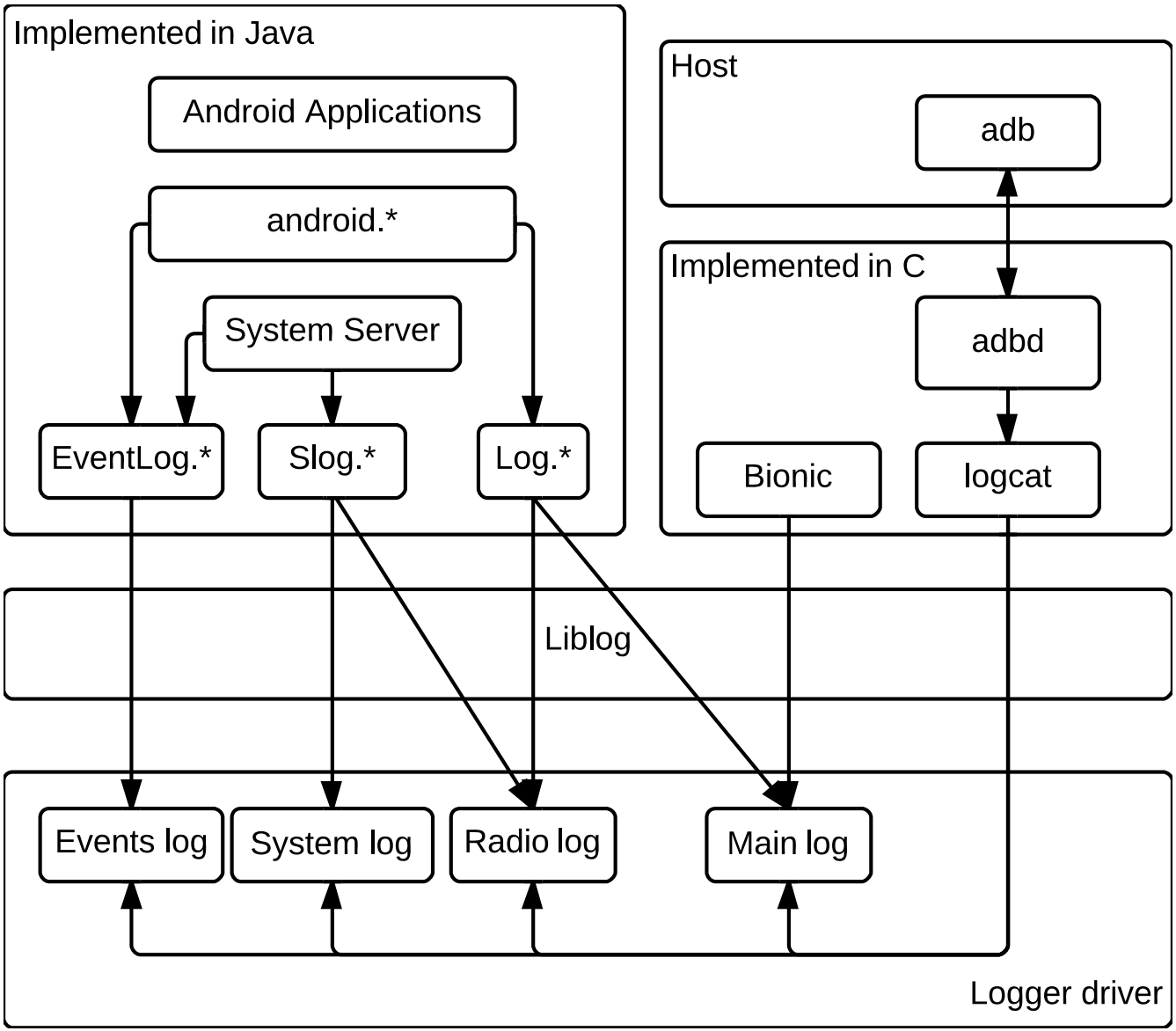
Logger

- Before Android 5.0
- Uses circular kernel buffers in RAM for logging data
- Each buffer - separate entry in `/dev/log` (Events, System, Radio, Main)
  - `logcat` displays the Main buffer by default
- Log, EventLog and Slog classes

Logging Messages

- Through liblog library
  - Logging from java classes
  - Used by `logcat`
  - Formatting and filtering
- Log message
  - Priority, tag and data for each event
  - Priority: verbose, debug, info, warn, error, assert
  - Tag: identifies the component that generated the message

Logging System



Logd

- From Android 5.0
- Logd daemon
- Centralized user-mode logger
- Addresses the disadvantages of circular buffers
- Integration with SELinux
  - Registers as auditd
  - Receive messages via netlink

Logd

- Uses 4 sockets
- `/dev/socket/logd` - control
- `/dev/socket/logdw` - write-only
- `/dev/socket/logdr` - read-only
- Unnamed netlink socket - SELinux

Logd

- Write log messages:
  1. `Log` class
  2. Liblog library
  3. `/dev/socket/logdw` socket
- Read log messages:
  1. `logcat`
  2. Liblog library
  3. `/dev/socket/logdr` socket

Paranoid Networking

- Standard Linux
  - Processes are allowed to create sockets and access the network
- Android
  - Restrict access to the network
  - Based on the group of the caller process
  - Group IDs
    - `AID_INET` - `AF_INET` and `AF_INET6` sockets
    - `AID_NET_RAW` - raw `INET` sockets
    - `AID_NET_ADMIN` - configuration of network interfaces and routing tables
    - `AID_NET_BT` and `AID_NET_BT_ADMIN` - Bluetooth

Outline

Binder History

- RPC mechanism
- Initially in BeOS (then bought by Palm)
- OpenBinder project
- OpenBinder developers working in Android team
- Android Binder does not derive from OpenBinder
  - Clean re-write of the same functionality
- OpenBinder documentation for understanding the mechanism
- Binder driver in the mainline from kernel 3.19

Binder

- Remote object invocation
  - Remote services as objects
  - Interface definition and reference to it
- Cornerstone of Android architecture
  - Apps talk to systems services
  - Apps talk to application services
- Developers don't use the Binder directly
- Use interfaces and stubs generated with the `aidl` tool
- Public API uses stubs to communicate with system services

Binder Driver

- Part of the Binder implemented in a kernel driver
- Character device
- `/dev/binder`
- `ioctl()` calls
- Transmit parcels of data (serialized) between entities

Outline

Android Framework

- On top of the native userspace
- `android.*` packages, System Services, Android Runtime
- Code in `/frameworks` directory in AOSP
- Key building blocks: Service Manager, Dalvik/ART, Zygote

System Services

- Form an object-oriented OS on top of Linux
- System Server
  - All components run in the `system_server` process
  - Many Java-based services/managers, 2 C-based services
  - Power Manager, Activity Manager, Location Manager, etc.
  - Surface Flinger, Sensor Service (C/C++)
- Media Server
  - `mediaserver` process
  - C/C++ code
  - Audio Flinger, Media Player Service, Camera Service

Dalvik

- Default before Android 5.0
- Java VM optimized for mobile architectures
  - Lower memory footprint
- Works with `.dex` files instead of `.jar` files
  - 50% smaller
- Incompatible with Java bytecode
- Register based, not stack based
- 16 bit instructions instead of 8 bit instructions (stack)
- Less instructions and higher execution speed

Dalvik

- Includes Just-in-Time (JIT) compiler
  - From Android 2.2
  - ARM, x86, MIPS
  - Profiles the applications at runtime
  - Translates segments of bytecode (traces) into machine instructions
  - Code runs directly on the CPU, not one instruction at a time by the VM
  - The rest of the bytecode interpreted by Dalvik
  - Performance improvements

Android Runtime (ART)

- Available from Android 4.4
- Default from Android 5.0
- Dalvik Executable format
- Ahead-of-Time compilation (AoT)
  - dex2oat tool
  - Translate the dex file into an executable for the target device
  - At installation time
  - Replaces JIT compilation and Dalvik interpretation
  - Installation takes longer
  - Executables occupy storage space
  - Additional verifications

ART

- Improved garbage collection
- Support for sampling profiler
- More debugging features
- More details in case of exceptions and crash reports

Zygote

- Daemon used to launch apps
- Parent of all processes
- Preloads in RAM all Java classes and resources needed by apps
- Listens to connections on its socket for requests to start apps
  - `/dev/socket/zygote`
- When it gets a request, it forks itself and launches the app

Zygote

- Copy-on-write (COW)
- Classes and resources are not modified, so all apps use them from Zygote
  - A single version of classes and resources in RAM
- The System Server is started explicitly by Zygote
- The PPID of all apps is the PID of Zygote

Outline

Service Manager

- Performs system service handle lookups
- The Yellow pages book of all system services
- A service must be registered to the Service Manager to be available
- Started by `init` before any other service
- Opens `/dev/binder` and becomes the Context Manager of the Binder
- Binder ID 0 = "magic object" = Service Manager

Service Manager

- System Server registers every service with the Service Manager
- Any component that wants to talk to a system service:
  - Asks the Service Manager for a handle
  - `getSystemService()`
  - Invokes the methods of the service using the handle
- Only to access system services
- Used by the `dumpsys` utility to obtain the status of the system services

Activity Manager

- One of the most important services in the System Server
- Handles activity lifecycle
- Sends intents
- Starts new components (activities, services)
- Obtains content providers
- Responsible with the Application Not Responding (ANR) messages
- Involved in
  - Permission checks
  - OOM adjustments for the Low-Memory Killer
  - Task management

Activity Manager

- Starts the Launcher (with `Intent.CATEGORY_HOME`)
- When an app is started from Launcher
  - Launcher's `onClick()` callback is called
  - Launcher calls the `startActivity()` from ActivityManager (through Binder)
  - ActivityManager calls `startViaZygote()` method
  - Opens socket to Zygote and asks to start the activity
- `am` command for invoking the functionality of the ActivityManager
- `isUserAMonkey()`

Package Manager

- Manages the `.apk` files in the systems
- API for installing, uninstalling, upgrading `.apk` files
- Works with files located in `/data/system/`
  - `packages.xml`
  - `packages.list`
- `system_server` and `installd` processes
- Resolves intents
- `pm` command for invoking the functionality of the PackageManager

Power Manager

- Control the power state of the device
- Handles WakeLocks
- Includes the WakeLock class
  - `acquire()`, `release()`
- Apps request WakeLocks from PowerManager
- All calls to the Power Management (kernel) go through PowerManager
- Can force device to go to sleep
- Set the brightness of the backlights

Bibliography

- Karim Yaghmour, "Embedded Android: Porting, Extending, and Customizing", Chapter 2
- Joshua J. Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, Georg Wicherski, "Android Hacker's Handbook", Chapter 2
- `https://wiki.linaro.org/LMG/Kernel/Upstreaming`
- `https://source.android.com/devices/tech/dalvik/`

Keywords

- Linux kernel
- WakeLocks
- Low-Memory killer
- Binder
- Ashmem
- Alarm
- Logger
- System Server
- Dalvik
- ART
- Zygote
- Service Manager
- Activity Manager
- Package Manager
- Power Manager