The Android Linux Kernel (Part 2): Core Kernel IPC & Processing Mechanisms

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

1. Recognize the two types of storage supported by Android Linux

2. Understand Android Linux’s local & remote communication mechanisms
Learning Objectives in this Part of the Lesson

1. Recognize the two types of storage supported by Android Linux
2. Understand Android Linux’s local & remote communication mechanisms
3. Know how Android Linux’s processes & threads mediate access to one or more processor cores
Android Linux Kernel: Local & Remote Inter-Process Communication (IPC)
Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services.

See [en.wikipedia.org/wiki/Inter-process_communication](en.wikipedia.org/wiki/Inter-process_communication)
Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services.

IPC is an essential part of mobile cloud computing (for clients)
Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services.

IPC is also an essential part of mobile cloud computing (& servers).
Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services

- It uses TCP/IP to access the Internet

See en.wikipedia.org/wiki/TCP/IP_model
Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services

- It uses TCP/IP to access the Internet
  - Optimized for LANs & WANs

Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services

- It uses TCP/IP to access the Internet
- It uses UNIX domain sockets for local communication on a device

See [en.wikipedia.org/wiki/Unix_domain_socket](en.wikipedia.org/wiki/Unix_domain_socket)
• Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services
  • It uses TCP/IP to access the Internet
  • It uses UNIX domain sockets for local communication on a device
    • Optimized for intra-host IPC

See [en.wikipedia.org/wiki/Unix_domain_socket](en.wikipedia.org/wiki/Unix_domain_socket)
- Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services
  - It uses TCP/IP to access the Internet
  - It uses UNIX domain sockets for local communication on a device
  - Its Binder driver supports non-standard message-oriented IPC on a device

See elinux.org/Android_Binder
Android’s local/remote inter-process communication (IPC) mechanisms mediate interactions between apps & system services

- It uses TCP/IP to access the Internet
- It uses UNIX domain sockets for local communication on a device
- Its Binder driver supports non-standard message-oriented IPC on a device
  - *Highly* optimized for intra-host IPC

See part 3 of this lesson on “Android Linux Kernel Extensions”
• The device driver framework runs in the kernel & coordinates access to hardware devices

See en.wikipedia.org/wiki/Device_driver
The device driver framework runs in the kernel & coordinates access to hardware devices, e.g.,

- Block-oriented devices
  - i.e., transfer data in “chunks”
The device driver framework runs in the kernel & coordinates access to hardware devices, e.g.:

- Block-oriented devices
- Character-oriented devices
  - i.e., transfer data “byte-by-byte”

The device driver framework shields other parts of the kernel & higher layers of Android from low-level hardware details. Hardware can thus be accessed portably, robustly, & securely.
- Programming & debugging device drivers is challenging!
• Programming & debugging device drivers is challenging!
• Requires low-level system architecture knowledge

See source.android.com/devices
Android Linux Kernel: Processes & Threads
The Android Linux kernel supports processes & threads.

See coltf.blogspot.com/p/android-os-processes-and-zygote.html
• The Android Linux kernel supports processes & threads
• Used to encapsulate app instructions & data efficiently, reliably, & securely on one or more processor cores
A process provides a unit of resource allocation & protection.
A process provides a unit of resource allocation & protection.

Each Android app typically runs in its own Linux process.

See developer.android.com/guide/components/processes-and-threads.html#Processes
• A process provides a unit of resource allocation & protection
• Minimize impact of app failures
• A process provides a unit of resource allocation & protection
• Minimize impact of app failures
• Ensure app data is private
A process provides a unit of resource allocation & protection
- Minimize impact of app failures
- Ensure app data is private
- Contain one thread by default

See developer.android.com/guide/components/processes-and-threads.html#Threads
• A process provides a unit of resource allocation & protection
  • Minimize impact of app failures
  • Ensure app data is private
• Contain one thread by default
  • Dispatches events to widgets & components in Android UI toolkit

See developer.android.com/guide/components/processes-and-threads.html#Threads
• Threads appear at multiple layers in the Android software stack
• Threads appear at multiple layers in the Android software stack
• We focus largely on kernel threads in this lesson
Threads appear at multiple layers in the Android software stack

- We focus largely on kernel threads in this lesson
- Upcoming lessons will cover threads in other layers

**Android Linux Kernel: Processes & Threads**

- **Application Frameworks**
- **Native Libraries**
- **Hardware Abstraction Layer**
- **Operating System Kernel**

**Threads**

- POSIX threads
- Java threads
- Android VM threads
• Threads provide units of execution for instruction streams that run on processor cores

See en.wikipedia.org/wiki/Thread_(computing)
Threads provide units of execution for instruction streams that run on processor cores

A Linux process has one thread by default
Threads provide units of execution for instruction streams that run on processor cores

- A Linux process has one thread by default
- Apps can create more threads via various API calls
Threads provide units of execution for instruction streams that run on processor cores
- A Linux process has one thread by default
- Apps can create more threads via various API calls
- Threads can run concurrently on one core
• Threads provide units of execution for instruction streams that run on processor cores
  • A Linux process has one thread by default
  • Apps can create more threads via various API calls
• Threads can run concurrently on one core
  • They can run in parallel on multiple cores
Threads provide units of execution for instruction streams that run on processor cores.

Each thread has a stack, a program counter, & other registers (unique resources).

See en.wikipedia.org/wiki/Thread_(computing) #Processes.2C_kernel_threads.2C_user_threads.2C_and_fibers
Threads provide units of execution for instruction streams that run on processor cores.

Open files & memory are shared across threads (shared resources).

See [en.wikipedia.org/wiki/Thread_(computing)](en.wikipedia.org/wiki/Thread_(computing)) #Processes, kernel_threads, user_threads, and_fibers
Android Linux kernel threads form the basis for the Java Threads in Android’s middleware infrastructure.

See developer.android.com/guide/components/processes-and-threads.html#Threads
Processes & threads consume non-trivial amount of system resources.
Programming multi-threaded apps is hard.
Programming multi-threaded apps is hard.

See en.wikipedia.org/wiki/Race_condition
• Programming multi-threaded apps is hard

See [en.wikipedia.org/wiki/Deadlock](en.wikipedia.org/wiki/Deadlock)
• Programming multi-threaded apps is hard
• Think deeply about how to program threads/processes
• Programming multi-threaded apps is hard
  • Think deeply about how to program threads/processes
  • Consult the Android documentation
Programming multi-threaded apps is hard

Think deeply about how to program threads/processes

Consult the Android documentation

& other online resources

Digital Learning Offerings

Douglas C. Schmidt (d.schmidt@vanderbilt.edu)
Associate Chair of Computer Science and Engineering,
Professor of Computer Science, and Senior Researcher
in the Institute for Software Integrated Systems (ISIS)
at Vanderbilt University.

Pearson LiveLessons Courses

- Concurrent Programming in Java
- Design Patterns in Java

Coursera MOOCs on Pattern-Oriented Software Architecture (POSA)

- Android App Development Coursera Specialization
- Spring 2014 Offering of Pattern-Oriented Software Architecture: Programming Mobile Services for Android Handheld Systems
- Spring 2013 Offering of Pattern-Oriented Software Architectures for Concurrent and Networked Software
Android Linux Kernel: Processes & Threads

- Java 8 concurrency & parallelism frameworks alleviate many complexities of writing multi-threaded apps

```java
List of URLs to Download
map(this::checkUrlCachedAsync)
map(this::downloadImageAsync)
flatMap(this::applyFiltersAsync)
collect(toFuture())
thenAccept(this::logResults)
```

```java
Parallel Streams
filter(not(this::urlCached))
map(this::downloadImage)
flatMap(this::applyFilters)
collect(toList())
```

See www.dre.vanderbilt.edu/~schmidt/DigitalLearning
End of the Android Linux Kernel: (Part 2) Core Kernel IPC & Processing Mechanisms