JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

מדיאטק הייטק
המכללה המובילה למחשבים והייטק
a matrix company

Android Internals No. 7170 – 40 Hours

Overview

Understand the inner workings of the Android Kernel and its various subsystems, from an architectural perspective, as well as through driver writing. In a top-down view, the course explains the components, and focuses on them using a sample device driver coded and expanded by the students. The students will become familiar with the Android source code (from Froyo to the KitKat), with thorough review of the sources, as well as detailed discussions of the various features introduced in each version.

This course allocates plenty of time for hands-on practice.
The hands-on exercises include:

- Creating a full-fledged Android character driver
- Manipulating Kernel structures like process control blocks, events and others

On Completion, Delegates will be able to

- Describe the architecture of the Android operating system
- Build and customize Android from source to x86, ARM, or other architectures
- Describe and customize the Android boot process
- Describe the functions and architecture of the Android Kernel
- Create a fully functional character device driver
- Explain the various Kernel subsystems, including the task scheduler, and virtual memory manager
- Understand how Android is similar and how it diverges from mainstream Linux

Who Should Attend

- Device Driver developers, embedded system developers planning to port or use Android and/or anyone interested in obtaining deeper insights into the workings of the Android Kernel. Also, Android developers who want to harness the abilities of the Android Native (external, non-Java) APIs.
- This course is **NOT** intended for user mode developers who wish to develop GUI applications or use the Android Java SDKs - but can be a great recommended follow-up for those who already do.

Prerequisites

- Knowledge of Android or Linux at a user level, and user mode programming. Familiarity with POSIX is highly recommended.

Course Contents

Introduction - The Android Architecture

Introduction to the Android architecture. We discuss the design and implementation of the various subsystems, at a modular "black box" level, without going into the source code level (yet).

- Android features
- Android vs. Linux vs. Embedded Linux
- Filesystem layout and directories
- The Runtime Environment
- The Frameworks
- Dalvik (Java) and ART
- Components
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers
- Version differences - 2.x and 3.0
- User-Mode and Kernel-Mode Architecture

Booting

System startup and initialization - Walk through the various stages, from boot loader through kernel startup to basic setup of user mode processes. Optionally, we also describe the modification of the boot loaders.

- The Boot loader
- Kernel Startup
- User mode init
- Services and Daemons

Exercises include:

- Creating a simple service and integrating it with the device boot sequence

Customizing Android

One of Android's strongest suits is its open source nature and relatively easy porting to almost any architecture. In this module we get familiar with the source code, obtaining it from the GIT repositories and compiling it with specific customizations. We also discuss porting Android to various architectures.

- Git'ing the source code
- Compiling and building
- Customizing
- Compiling and tweaking for x86, x64, or ARM

Exercises include:

- Obtaining the source code of the latest Android system, compiling and building it
-

Crash course in application development

We walk through a simple Android application - working through the Android SDK, the Dalvik Java environment, and the basic frameworks. This module is optional, in case the target audience are new to Android, and is not meant to be comprehensive - only to cover the key concepts, and map them to the underlying native APIs that implement them. Otherwise, it can be skipped. Topics include:

- Basic application structure - Packages and Manifests
- Text Views
- Buttons
- Basic Layouts
- Menus
- Dialogs
- Events and Notifications

Exercises include:

- Compiling and building a sample Android "Hello World" application

The NDK

For developers who need to escape the Virtual Machine, Android offers the Native SDK - or the NDK. For performance critical regions, going into native mode is a necessity, not a luxury. We explain the basic ideas of the NDK, its bridging from Java, and sample from its APIs.

- Why go Native? Introduction to NDK
- Bionic - Android's LibC
- Accessing Native Calls - JNI
- NDK Samples

Exercises include:

- Compiling and building a sample NDK application

Process and thread internals

We dive deeper into the native environment, explaining how processes and threads are actually implemented in Android.

- Processes and threads - in depth
- Creating processes - the clone() system call
- Maintaining processes/threads - the task_list
- The Process/Thread control block - struct task_struct
- Threading
- Priorities & scheduling
- Maintaining virtual memory

Interfacing with the kernel

- System Calls - In depth:
    - System call modes - Interrupt gates vs. SYSENTER/SYSCALL
    - Adding system calls to the Kernel
    - Calling system calls from Kernel Mode
    - System call tracing using API

Kernel Survival Guide

Going deeper into the Kernel, we explain the basics of an Android Kernel module.

- Writing code for Kernel mode
- Dos and Donts of Kernel mode
- Plugging into the Kernel
- Sample Kernel Module

Exercises include:

- Creating a simple kernel module

Creating a basic device driver

Creating a basic character device driver, including:

- System calls from the driver's perspective
- Device Major & Minor numbers
- register_chrdev
- udevd and hotplug
- Interfacing with sysfs (/sys directory)

Exercises include:

- Creating a simple driver

Android-isms

Android is, at the kernel level, almost identical to the common Linux kernel. There are, however, idiosyncrasies and modifications, which optimize the kernel to the mobile environments - low memory conditions, power management considerations, and more. In this chapter, we dig deep into these modifications, by examining each in detail, at the source code level. Specifically, we discuss:

- ASHmem - Android's Anonymous Shared Memory mechanism
- PMem - Physical contiguous memory for non scatter/gather capable hardware
- Low memory killer - Allowing customized Linux OOM behavior from user space
- Wakelocks - and the enhancements to Power Management
- Android Logging - /dev/log/* and the magic behind the logcat command
- The RAM Console - Used in Android to save Panic data across reboot
- /dev/binder - the implementation of OpenBinder and the underlying mechanism supporting AIDL
- Timed GPIO/Output

Android Security

The Android security infrastructure and model.

- Security Policies
- Private and Public
- Permissions
- Service Hooks
- Protected APIs