

KOENIG SOLUTIONS • COURSE OUTLINE

Android AOSP Development (Android ATC)

Learn to build, customize and extend AOSP, from kernel and build system internals to HAL, debugging and native feature development.

4 Days

DURATION

7

MODULES

54

TOPICS

34

LAB EXERCISES

COURSE OVERVIEW

This course covers the Android operating system and AOSP, starting with the Linux kernel, AOSP directory structure, kernel changes, and the Android boot process. It then moves to customizing the Android system, build system, security policies, bootloaders, ADB debugging, and AIDL/HIDL concepts. It continues with extending AOSP through new features, the native layer, SELinux policies, and HAL modifications, including crash debugging and media buffer handling. It concludes with advanced build techniques, filesystem layout, daemon services, HAL-framework integration, and CTS/VTS compatibility testing.

WHO THIS COURSE IS FOR

Embedded Systems Engineers

Platform Developers

Kernel Developers

Device Driver Engineers

Firmware Engineers

Android System Engineers

LEARNING OBJECTIVES

- Explain the Android architecture from kernel to application framework
- Describe the AOSP directory structure and build system
- Apply kernel configuration and cross-compilation for Android devices
- Configure and customize system applications, settings and build IDs
- Implement SELinux policy and HAL modifications for custom devices
- Debug managed, native and kernel components using ADB and related tools
- Analyse crash traces and buffer flow between media components
- Assess compatibility using CTS and VTS test suites

7 Modules, 54 Topics

DAY 1

Introduction to Android and AOSP

01

Theory

OVERVIEW OF THE ANDROID OPERATING SYSTEM

- History and evolution of Android
- **Android architecture and components:** Layers from hardware abstraction to application framework

INTRODUCTION TO THE LINUX KERNEL

- Kernel features and development process
- Legal constraints with device drivers
- Kernel user interface (/proc and /sys)
- Kernel configuration
- Native and cross-compilation

INTRODUCTION TO AOSP

- What is AOSP? How it enables custom Android development.
- Differences between AOSP and Android
- AOSP ecosystem and community

AOSP DIRECTORY STRUCTURE

- Key directories and their purposes

CHANGES INTRODUCED IN THE ANDROID KERNEL

- Functional changes introduced by Google
- Additions to the kernel
- Mainline kernel status of patches

DETAILED ARCHITECTURE OF ANDROID

- From hardware level to application level

BOOTING STAGES OF ANDROID

- In-depth understanding of the booting process (bootloader), kernel, and user-space initialization

Lab 1.1

3 Lab Exercises

Lab 1.2

4 Lab Exercises

Lab 1.3

4 Lab Exercises

02

Theory

CUSTOMIZING THE ANDROID SYSTEM

- Modifying system applications
- Customizing system settings (Customizing the build ID)

INTRODUCTION TO CUSTOM ROMS

- Building and modifying custom ROMs

ANDROID BUILD SYSTEM

- Makefile architecture and functions
- Adding a new device to the build system

SECURITY AND PERMISSIONS IN AOSP

- Understanding and modifying security policies

BOOTLOADERS

- Bootloader examples and fastboot specifications

DEVELOPING AND DEBUGGING WITH ADB

- File transfers, package installation, logging, and debugging

AIDL AND HIDL CONCEPTS

- Introduction and differences
- Implementation in Android

Lab 2.1

3 Lab Exercises

03

Lab 2.2

CUSTOMIZING SYSTEM APPLICATIONS

Modify system apps and system settings · Customize boot screens and build IDs · Rebuild and test on Raspberry Pi

ADDING A NATIVE LIBRARY

Create and integrate an external library · Test on Android builds

Lab 2.3

6 Lab Exercises

04

Theory

ADDING NEW FEATURES TO AOSP

- Implementing, testing, and debugging new features

ANDROID NATIVE LAYER

- Understanding Android runtime components
- Exploring hardware abstraction and media framework

SELINUX POLICIES

- Overview and purpose
- Modifying SELinux policies for custom devices

HAL AND HAL MODIFICATIONS

- Hardware Abstraction Layer overview
- Customizing and extending HAL

05

Lab 3.1

IMPLEMENTING A NEW FEATURE

- Add and test a simple new feature in AOSP

USING RASPBERRY PI

- Boot Android and troubleshoot issues on Raspberry Pi

DEVICE DEVELOPMENT

- Add a new device to the AOSP build system
- Explore daemons handling hardware components

Lab 3.2

5 Lab Exercises

Lab 3.3

4 Lab Exercises

06

Theory

ADVANCED BUILD SYSTEM TECHNIQUES

- Variables and compilation steps
- Customizing the build environment

ANDROID FILESYSTEM LAYOUT

- Software components installation and importance

ADVANCED DEBUGGING WITH ADB

- Networking, remote commands, and system logs

HAL, FRAMEWORK, AND APPLICATION INTERCONNECTION

- Understanding the data flow and integration

DAEMON SERVICES

- Role of daemon services in Android
- Managing and customizing daemon services

CTS AND VTS

- Compatibility Test Suite overview
- Vendor Test Suite and its importance

07

Lab 4.1

ADVANCED SYSTEM CUSTOMIZATION

Modify “About” info and boot parameters

ADVANCED FEATURE INTEGRATION

Add a feature requiring hardware abstraction

FINAL TESTING AND DEBUGGING

Full system validation on Raspberry Pi

Lab 4.2

5 Lab Exercises

Practise What You Learn

MODULE	ACTIVITY
Lab 1.1	■ Setting Up the Development Environment: Installing Android Studio and SDK; Cloning the AOSP repository; Using Android-specific tools ■ Hands-on with AOSP Build: Practice building and running AOSP; Troubleshooting common build issues ■ Using the Android Emulator: Compile and boot an Android Kernel; Extract patches from the Android Kernel
Lab 1.2	■ Exploring Sandboxing and App Communication: Create a simple Android app that requests permissions (e.g., location or camera) ■ Analyze app isolation by inspecting /data/data directory ■ Use Binder IPC to fetch data from a system service, such as LocationManager ■ Inspect the interaction using adb logcat and analyze the logs for Binder transactions
Lab 1.3	■ Hands-on with Android-Specific Tools: adb - Use adb devices to list devices ■ adb - Transfer a file to a connected device using adb push ■ adb - Inspect logs using adb logcat ■ logcat - Capture and filter logs for specific tags
Lab 2.1	■ Hands-on with Android-Specific Tools (Continuation): fastboot - Flash a custom boot image using fastboot flash boot <boot.img> ■ systrace - Generate a system trace to analyze app performance ■ perf - Use the perf tool to analyze kernel-level performance
Lab 2.3	■ Debugging the Android Platform User Space - Managed Components Debugging: Build a simple Android app with a deliberate error ■ Managed Components Debugging: Use adb logcat and Android Studio debugger to identify and fix the error ■ Native Debugging: Write a native library in C/C++ that causes a segmentation fault ■ Native Debugging: Debug using gdbserver and analyze the core dump ■ Kernel Debugging: Boot a custom kernel on an emulator ■ Kernel Debugging: Use kgdb or inspect kernel logs via dmesg
Lab 3.2	■ Crash Debugging and Trace Analysis: Simulate a crash in a native application by triggering a segmentation fault ■ Analyze the crash using: adb logcat for immediate stack traces ■ Analyze the crash using: Tombstone files (/data/tombstones) for detailed debugging ■ Use symbols from .so files to resolve native traces ■ Experiment with tools like Crashlytics to capture managed crashes

MODULE	ACTIVITY
Lab 3.3	■ Media Buffer Passing Between Components: Implement a basic app that captures video using the Camera API ■ Implement a basic app that encodes video using MediaCodec ■ Analyze the use of shared memory (Ashmem) for buffer exchange ■ Inspect logs to trace buffer flow through services like AudioFlinger or MediaCodec
Lab 4.2	■ Deep Dive into IPC and Shared Memory: Implement a basic Android Service ■ Add an AIDL interface for inter-process communication ■ Build a client app to interact with the service ■ Experiment with shared memory: Transfer large data buffers and analyze performance ■ Inspect Binder transactions using the binder debugfs interface

WHAT IS INCLUDED

Everything You Need to Succeed

- Koenig course material for every module
- Hands-on exercises throughout the course
- Delivery by certified instructors and subject matter experts
- Course completion certificate from Koenig

ABOUT KOENIG

Who We Are

Established in 1993, Koenig Solutions is a globally recognised training provider with 35+ technology partnerships, 5,000+ training programmes and a network of certified instructors and subject matter experts, supporting individuals and enterprise teams worldwide.

FLEXIBLE DELIVERY

Learn the Way That Suits You

Classroom Training

Live Online Classes (ILO)

Fly-Me-A-Trainer (FMAT)

Flexi (Self-Paced)

1-on-1 Training

Ready to Get Started?

Talk to a learning advisor about schedules, group training and customised delivery.

Visit koenig-solutions.com

✉ info@koenig-solutions.com