G3 - Android Internals

Android Frameworks and HAL Implementation

Objectives

- Explore the Android source code architecture
  - The Android init process
  - System services
  - The Android Binder
  - Android security and SELinux
  - The Android Application Framework
  - The Android Hardware Abstraction Layer
  - The Android Multimedia Framework and OpenMAX
- Understand the static and dynamic framework structure
  - Class structure
  - Split between Java and C++ code
  - Data flow through the frameworks
  - Control structure of the frameworks

Labs are conducted on the Android emulator
We use the last released AOSP (Android Open Source Platform) version.
For on-site trainings, if suitable Linux workstations are not available, we provide virtual machine images for VirtualBox; in all cases the requisite is a recent 64bit PC (at least 4 cores) with 64Gb of RAM (32Gb may work but is not recommended) and 600Gb of free disk space.

Who should attend this course?

- Engineers that must work on the Android port on a new platform
  - Writing the HAL for a new board
  - Debugging the HAL and Android frameworks
  - Expanding the Android platform for specific usages

Prerequisite

  - Good Linux kernel and driver programming experience
  - Android installation knowledge
  - Basic knowledge of the structure of an Android application
  - Good C++ and Java programming skills
  - Basic Java knowledge

Course environment

  - Printed course material (in English).
  - One Linux PC for two trainees (64GB RAM, 1TB free disk, 8 cores).
  - Debug on Android emulators (Goldfish or Cuttlefish)

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

Android Architecture Overview

- Linux and Android
- Android Licensing

First Day

The Android Linux kernel enhancements

- Accessing the kernel source code
- The Android-specific kernel drivers
  - Ashmem
  - Logger
  - Low_memory_killer
  - Timed_output
  - Timed_gpio
  - Buttons and Keypad management
- The Android GKI kernels
- The Android Kernel debugger

Exercise: Setting up the build environment and launching the build

The Android Build system

- The Android code base
- Building Android
  - The Android build environment
  - The Android Emulators (Goldfish and Cuttlefish)
  - The Android build system
  - The Android.mk files
- Adding new components to the build system
  - Java components
  - Native components
  - Applications
- Creating a new Android platform
- Converting to Soong
  - Blueprint files
- Android Treble
  - Android Partitions
  - Android Shared System Image (SSI)
  - Interface Enforcement
- Debugging

Exercise: Creating and compiling a new Android platform

Android System Initialization

- Android properties
- The Android initialization
  - Android Initialization and Device management
  - The Android initialization language
  - The Android boot sequence
  - The “zygote” process
- Security Enhanced Android

    - Security in Android and SELinux
    - Toolbox commands
    - Security and the build system
    - SEPolicy files

Android Native Interface

- The bionic C library
    - Why a new C library
    - The bionic Android-specific features
    - What is missing in bionic
- Adding native components
    - Adding native executables
    - Defining Java methods in C++
    - JNI for Android
- Platform interface for native components
    - Accessing system properties
    - Accessing the Android log system
    - Interacting with daemon services
- The Android NDK and JNI
- SWIG

Exercise: Creating a new native component

Second Day

The Android Binder

- The Android Binder architecture
    - Why a new IPC mechanism
    - Android as a massively distributed system
    - The Binder in action
    - The Binder kernel driver
- Binder implementation
    - The AIDL language
    - The AIDL tool
    - Binder Java classes
    - C++ binder implementation classes
- Writing Services
    - Standard Java services
    - Services and the Binder
    - Service Binding
    - Stable AIDL
- Binder implementation
    - Binder Java classes
    - Reference counting in C++ Android frameworks
    - C++ binder implementation classes
    - Implementing a C++ Service
    - The AIDL NDK backend
- System services
    - What is a system service
    - Static and context-dependent services
    - Structure of a system service
    - Adding a new system service
    - The system ServiceManager process

Exercise: Coding a system service

Third Day

The Android Power Manager

- Android Power Management
- The Driver API
- The user-mode API
- The Java API

The Hardware Abstraction Layer

- Why a HAL?
- Conventional and Legacy HALs
  - Architecture
  - Main HAL components
  - The legacy Sensor HAL
- HIDL HALs
  - HAL Types
  - The Hardware Interface Definition Language
- Migrating from HIDL to AIDL

Exercise: Create a simple HAL component

Fourth Day

The Android Sensors Framework

- Sensors in Android
  - The sensor types
  - The Sensor Manager
  - Accessing Sensors
- Framework Architecture
  - Sensor discovery
  - Sensor Calibration
- The Location Manager and Geocoding

The Android Multimedia Framework

- Multimedia in an Android device
  - Data formats and File formats
  - Codec and Demux
- Multimedia for Applications
  - Audio and video playback (MediaPlayer class)
  - Audio and video capture (MediaRecorder class)
- Framework Architecture
  - General framework architecture
  - General data and control flows
  - The MediaPlayer service layer
  - Stagefright and OpenMAX

Exercise: Implementation of an mp3 playback service

Stagefright and OpenMAX for Android Multimedia

- Multimedia in an Android device
- StageFright in the Android Media Framework
  - The legacy OpenCore framework
  - The Stagefright class structure

- OpenMAX Overview
  - The Khronos Group
  - OpenMAX/DL: the Development Layer
  - OpenMAX/IL: the Integration Layer
  - OpenMAX/AL: the Application Layer
  - OpenMAX and OpenSL/ES
- OpenMAX in the Android Media Framework
  - Interface between Android and OpenMAX
  - The OpenMAX/IL Architecture
- Integration of OpenMAX/IL in Stagefright
  - Component registration
  - Component configuration
  - Component Quirks
- The OpenMAX/IL Bellagio implementation
  - OpenMAX/IL LGPL implementation of the core
  - Sample implementation of components

Fifth Day

The Surface Flinger

- Overview
- The Hardware composer

The Android Audio Manager

- Audio routing
- Architecture of Audio manager
  - Audio system
  - Audio policy manager
  - Audio policy service
- Control flow
  - Playback and recording control
  - Time source generation

The Audio Flinger

- Output/input audio flow
  - Buffer management
- Audio track
  - Overview
  - Track live cycle
  - Data flow
- Audio mixer:
  - Overview
  - Mixer life cycle
  - Fast mixer
  - Resampling
  - Volume

The HAL Audio components

- Audio policies
  - Audio Policies and policy devices
  - HAL Audio policy provided services
  - Use from the audio policy manager
  - Use of audio services by the HAL audio policy

    - Output duplication
    - Suspend/resume
- Audio devices
    - Audio device classes
    - Data flow
    - Interaction with audio track and record
- Audio streams and effects