G4 - Portage Android et structure interne

Portage du système Android et dea Frameworks

Objectifs

- Comprendre l'architecture du système Android.
- Apprendre à utiliser le gestionnaire de version GIT pour télécharger et gerer les sources d'Android
- Apprendre à installer Linux sur votre plateforme et à créer un BSP
- Découvrir la version Android du noyau Linux.
- Nouveaux drivers etIPCs
- Gestion de consommation
- Explorer L'architecture du système Android
- L'initialisation d'Android
- Les services système
- Le Binder Android
- La couche HAL (Hardware Abstraction Layer)
- Les Framework Multimedia et OpenMAX
- Apprendre à installer Android sur une plateforme qui supporte déjà Linux

  Labs are conducted on i.MX6 or i.MX8 boards

  We use the last open source version of Android, as available on the board.

  For on-site trainings, if suitable Linux workstations are not available, we provide virtual machine images for VirtualBox; the only requisite is then a recent 64bit PC with at least 8Gb of RAM and 100Gb of free disk space.

Who should attend this course?

- Engineers that must install an Android platform on a new board
- Porting the Linux kernel from a supported SoC
- Adapting the Android Frameworks to the board hardware
- Tailoring the Android System
- Connecting hardware graphics accelerators to the Android framework

Prerequisite

- Embedded Linux experience (see our cours D1 - Linux embarqué avec Buildroot et Yocto)
- Good Linux kernel and driver programming experience (see our cours D3 - Drivers Linuxcourse)
- Good C++ and Java programming skills (see our cours L3 - C++ embarquéand cours L4G - Java pour Androids)

Course environment

- Printed course material (in English).
- One Linux PC for two trainees.
- One target platform (dual Cortex/A9) for two trainees.
- Lauterbach JTAG probe for debug.

Environnement du cours

- Cours théorique
- Support de cours imprimé et au format PDF (en anglais).
- Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Activités pratiques
- Les activités pratiques représentent de 40% à 50% de la durée du cours.
- Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
- Exemples de code, exercices et solutions
- Un PC (Linux ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
- Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.

- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

First Day

Android Overview

- Linux
- Android
- Android licensing

The GIT and REPO distributed source management system (reminders)

- Installation and general usage.
- Creating and using a local repository
- Cloning a remote repository
- Working with branches
- Creation of a new branch
- Merging branches
- Team functions
- Creating configuring and managing a public repository
- Working with patches
- The repo tool
- The manifest file
- Downloading the Android source tree
- Adding projects to the manifest
- The Android development process
- Android code lines
- Submitting changes through Gerrit

The Android Linux kernel enhancements

- Downloading source code
- Adapting the Android-specific kernel drivers
- Alarm
- Ashmem
- Logger
- Low_memory_killer
- Timer_gpio
- Timed_output
- Buttons and Keypad management
- The Android Binder architecture
- Why a new IPC mechanism
- The Binder in action
- The Binder kernel driver
- Android Power Management
- The Linux Power Management architecture
- Android Wake Locks
- The Power Management driver
- The Android Kernel debugger
- Configuring an Android Linux kernel
- Building the kernel
  Exercise : Configuration and build of the Android kernel for the target board
  Exercise : Checking the first phases of kernel boot

Second Day

The Android Build system

- The Android code base
- Building Android
- The Android build environment
- The Android build system
- The Android.mk files
  Exercise : Compiling a single component
- Creating a new Android platform
- Declaring a new vendor
- Creation of platform-specific parameter files
- Choosing platform-dependent compilation options
  Exercise : Creating and building a new Android platform

Android Application Structure

- Structure of an Android Application
- Android application components
- Activity
- Service
- Broadcast receiver
- Content provider
- Manifest file
- Application components declaration
- Permissions
  Exercise : Hello world application

Android System Initialization

- Android properties
- Automatic properties
- Default properties
- Persistent properties
- The Android initialization
- Structure of the init process
- The Android initialization language
- The Dalvik zygote process
  Exercise : Modify the init process to handle (simulated) firmware signature check

The Android BSP components

- The Dalvik Java virtual machine
- The Dalvik machine structure
- The Dalvik bytecodes
- The Dalvik JIT compiler
- Porting the Dalvik interpreter
- Porting the JIT compiler
- Adding native components
- Adding native executables
- The Android NDK
- Defining Java methods in C or C++
- JNI for Android
- Using SWIG
  Exercise : Add a native Linux system component to the Android BSP

Third Day

The Android Native Framework

- The bionic C library
- Why a new C library
- The bionic Android-specific features
- What is missing in bionic
- Compiling against bionic

- Creating and using shared libraries
  Exercise : Create a simple program and shared library using the bionic library
- WebKit
- The Media Framework
- The OpenCORE PacketVideo platform
- Supported audio, video and still formats
- Hardware and software codec plug-ins
- SQLite
- FreeType
- SSL
- OpenGL/ES

The Android Application Framework

- The Core platform services
- Activity Manager
- Package Manager
- Window Manager
- Resource Manager
- Content Providers
- View System
- The system services
- What is a system service
- Telephony Service
- Location Service
- Bluetooth Service
- WiFi Service
- USB Service
- Sensor Service
- Power Service
- The Android binder
- Writing services in C++
- The binder's C++ interface
- Adding a new system service
  Exercise : Creating a system service in C++

Fourth Day

The Android Native Servers

- Android Renderscript layer
- The Android framework rendescript API
- The Reflected layer mapping rendescript code to Java classes
- Renderscript code
- The renderscript graphics and compute engine
- Interfacing the Renderscript engine with hardware accelerators
- The Surface Flinger
- The Binder interface
- OpenGL/ES interface
- Using hardware accelerators and composers
- Double buffering using page-flip from the frame buffer
- The Audio Flinger
- Handling output devices
- Handling audio routing

The Hardware Abstraction Layer

- Why a HAL?
- The Acme Component-oriented Architecture Definition Language
- Defining HAL components in Acme
- Loading and using HAL component
- The standard HAL components
- Graphics
- Audio
- Camera
- Bluetooth
- GPS
- Radio (RIL the Radio Interface Layer)
- WiFi
  Exercise : Create a simple HAL component

Fifth Day

OpenMAX for Android Multimedia

- Multimedia in an Android device
- Data formats and File formats
- Codec and Demux
- The Android Multimedia Framework
- OpenCORE the initial Android Media Framework
- OpenCORE architecture
- Stagefright
- 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

The Linux BSP

- Linux BSP architecture
- Overall structure
- The ARM BSP
- The Linux build system
- Linux kernel memory usage
- Defining and initializing the board
- Linux kernel debugging
- Debugging with a JTAG probe
- Debugging with traces
- Debugging with kgdb and kdb
  Exercise : Debugging the first steps of kernel startup
  Exercise : Debugging kernel modules and drivers
- The Linux driver model
- Kernel objects
- Devices, Classes and Drivers
- Hotplug events
- Power Management in drivers