D1S - Embedded Linux with Ac6 System Workbench

Implementing Linux on Embedded Systems

Objectives

- Understanding the architecture of the Linux system
- Learn how to install Linux on your hardware and create a BSP
- Explore the Linux system architecture
- Booting Linux
- Initializing the system
- Install existing packages on the target
- Learn how to install Linux on flash chips

All labs are conducted using the System Workbench for LinuxIDE.

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version for face-to-face courses.
  - Online courses are dispensed using the Teams video-conferencing system.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - For remote trainings:
    - One Online Linux PC per trainee for the practical activities.
    - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
    - QEMU Emulated board or physical board connected to the online PC (depending on the course).
    - Some Labs may be completed between sessions and are checked by the trainer on the next session.
  - For face-to-face trainings:
    - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - One PC for two trainees when there are more than 6 trainees.
  - For onsite trainings:
    - An installation and test manual is provided to allow preinstallation of the needed software.
    - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Prerequisite

- Good C programming skills
- Knowledge of Linux user programming (see our D0 - Linux user mode programmingcourse)
- Preferably knowledge of Linux kernel and driver programming (see our D3 - Linux Driverscourse)

Target Audience

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

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Day

Introduction to Linux

- Linux history and Version management
- Linux system architecture
  - Processes and MMU
  - System calls
  - Shared libraries
- Linux components
  - Toolchain
  - Bootloader
  - Kernel
  - Root file system
- Linux distributions
- Linux packages
- The various licenses used by Linux (GPL, LGPL, etc)

Linux tools for embedded systems

- Boot loaders (UBoot, Redboot, barebox)
- Optimized libraries (eglibc, uClibc)
- Toolchains
- Embedded GUIs
- Busybox
- Embedded distributions
  - Commercial
  - Standard
  - Tools for building custom distributions

Introduction to System Workbench

- Overview
  - Eclipse
  - Kernel and modules
  - Platforms and Root file-systems
- The build system architecture
  - Building individual packages
  - Building platforms
  - Building Root file-systems

Exercise: *Building a root file system a pre-defined platform template*

Developing applications with System Workbench

- Creating a Linux program
- Creating a library
  - Static library
  - Shared library
- Debugging on the target
  - Using an SSH connection
  - Debugging shared libraries

Exercise: *Create a small program, with a custom shared library, and debug it on the target*

Using U-Boot

- Introduction to U-Boot
- Booting the board through U-Boot
  - Booting from NOR
  - Booting from NAND
  - Booting from eMMC
- U-Boot environment variables
  - User-defined variables
  - Predefined variables
  - Variables substitution
- The U-Boot minimal shell
  - Writing scripts in variables
  - Executing scripts
  - Using variables in scripts: the set-script pattern
- U-Boot main commands
  - Booting an OS
  - Accessing flash chips
  - Accessing file systems (NFS, FAT, EXTx, JFFS2&)
- The full U-Boot shell
  - Script structure
  - Control flow instructions (if, for&)
- Booting Linux
  - Linux kernel parameters
  - The Linux startup sequence

Exercise: *Writing a script to configure the network and pass this configuration to the Linux kernel*
Exercise: *Booting the board on NFS, using pre-existing images*
Exercise: *Writing scripts to choose between boot from flash or from the network*

Second Day

Building U-Boot

- Building and installing U-Boot with its native build system
- Building U-boot with System Workbench

Exercise: *Configuring and building u-boot with its native build system*
Exercise: *Building u-boot from System Workbench*

Building the kernel

- The Linux build system
  - Downloading stable source code
- Getting a tarball
- Using GIT
  - Configuring the kernel
  - Compiling the kernel and its modules
- Modules delivered in-tree

- Out-of-tree modules
  - Installing the kernel and the modules

Exercise: *Configuring and compiling a target kernel for the target board with the kernel build system*

- Kernel projects
  - Creating a kernel project
  - Selecting the architecture and configuration
  - Customizing the configuration
  - Compiling the kernel

Exercise: *Configure and compile the kernel in the platform*

- Module projects
  - Creating a module project
  - Linking it to a kernel project
  - Creating and building modules

Exercise: *Add and configure an external module*

Exercise: *Exercise: Configuring and compiling a target kernel for the target board with System Workbench*

Building packages

- Packages
  - Tools to build packages (gcc, Makefile, pkg-config&)
  - Autotools
  - Cross-compiling a package with autotools
- The all-in-one applications
  - Busybox, the basic utilities
  - Dropbear: encrypted communications (ssh)
- Automatically starting a program at boot
  - Initialization systems (busybox init, system V init)

Exercise: *Cross-compiling an autotools-based package*

Creating a Linux Platform

- Creating a platform project
  - Importing a pre-configured platform
  - Creating a platform from scratch
- Configuring the platform
  - Source and installation directories
  - Link to a target Rootfs
  - Build configurations

Exercise: *Create and configure a minimum platform from scratch, using library packages*

- Populating the build environment
  - Import packages in the build environment
  - Build individual packages
  - Build the whole platform

Exercise: *Build the platform, manually building some packages*

- Adding packages to a platform
  - From a library
  - From an existing Eclipse project

Exercise: *Add the previously developed application to the platform*

- Creating a new package
  - Specifying the source
  - Patching the official sources
  - Adding package-specific resources
  - Adding package configuration directives

Exercise: *Add a new open-source package to the platform*

Third Day

Embedded file systems

- Storage interfaces
  - Block devices
  - MTD
- Flash memories and Linux MTDs
  - NOR flashes
  - NAND flashes
  - ONENAND flashes
- The various flash file system formats
  - JFFS2, YAFFS2, UBIFS
- Read-only file system
  - CRAMFS, SQUASHFS
- Standard Linux file systems
  - Ext2/3/4, FAT, NFS
  - Ramdisks and initrd
  - Creating an initramfs
  - Booting through an initramfs
- Choosing the right file system formats
- Flashing the file system

Creating a root file system

- Manually building your root file system
  - Device nodes, programs and libraries
  - Configuration files (network, udev, &)
  - Installing modules
  - Looking for and installing the needed libraries
  - Testing file system consistency and completeness
- Package management
  - ipkg

Exercise: *Manually creating a minimal root file system using busybox and dropbear*

- Creating a rootfs project
  - Creating the rootfs structure
  - Add files to the base structure
- Edit standard configuration files
  - File systems
  - Initialization
  - Starting applications

Renseignements pratiques

Duration : 3 days
Cost : 2070 € HT