Hands-on course , 5 day(s)
Ref : BLE

Participants

Embedded systems architects and developers in charge of Linux or Linux application deployment on an embedded system.

Pre-requisites

Basic Linux administration and C development know-how.

Next sessions

Embedded Linux, BSP and uBoot

OBJECTIVES

This training will bring you knowledge of the stages to add a new BSP to uBoot and Linux. You will learn how to build an embedded systems toolchain and the embedded root filesystem. Major issues related to the Linux kernel will be pointed and finally you will embed the graphical libraries and Linux systems utilities.

Workshop

Every step of the training session is immediately applied as a case study on an embedded ARM board with a touch screen to test graphical developments.

1) The cross development tools

- Overview of an embedded system and of the Linux kernel architecture.
- Cross development tool chain, gcc cross compiler, C libraries, glibc and uClibc, GNU debugger, GNU ELF tools.
- Embedded development tools, QEMU, Buildroot, Busybox and Scratchbox

2) The universal Boot loader: uBoot

- uBoot project overview. A walk through the source code. Supported architectures. Basic functionalities.
- The uImage format for booting uBoot Images.
- Configuration, compilation and installation in a QEMU sandbox for testing.
- Development of a standalone program using uBoot as BIOS.
- uBoot BSP. Adding a new SOC and a new board in the uBoot BSP tree.

Workshop
Add a new command to uBoot and test uBoot inside QEMU, generate a new BSP for uBoot and develop a simple stand alone program using uBoot as BIOS.

3) Linux kernel

- Licenses implications and kernel modules development. Development cycles.
- Kernel development tools, quilt, GDB, GIT, LTT. Configuration tool Kbuild.
- The Linux boot process.
- Devices drivers. The Linux driver framework and standard drivers.
- The Linux BSP. Adding a new board to Linux.
- Specific embedded systems drivers MTD drivers, CAN, SPI and I2C drivers.

Workshop
Modify the kernel tree to add a new driver to the kernel tree and generate a patch formatted for the LKML. Develop a character driver outside of the kernel tree.

4) Root File system

- Creation of a rootfs. A tiny root file system, back to UNIX fundamentals and the init program.
- Manage users on an embedded system with busybox.
- Dynamic libraries or static programs. Choosing a root file system architecture.
- Building a rootfs as CPIO or as EXT2 file system.
- Creating JFFS2, UBIFS or YAFFS file systems.

Workshop
Create rootfs from scratch using busybox and test it on a real ARM target.Use buildroot to add new applications.Add your own application developed using SCRATCHBOX.Test the buildroot generated rootfs.

5) Limits of Linux Embedded

- Industrial realtime application.
- Power management.
- Embedded interfaces.
- Complete embedded framework.
- Debugging. Using QEMU to debug and embedded system.

Workshop
You will modify the Linux boot-logo using standard Linux graphical tools. Test a hard realtime solution. Debug an application on the ARM target.