Training_

Hedera Innovations Limited has formed partnership with Feabhas Ltd to deliver the following training courses on Embedded Linux.

Training courses

EL-503

Developing for Embedded Linux

Length: 5 days

Course Description:

Embedded Linux is rapidly becoming an Operating System of choice for many embedded developers. According to the LinuxDevices.com website, the significant reasons are that the source code is available, there are no runtime royalties and it is a robust reliable operating system which has excellent networking support.

This course, like other Feabhas courses, teaches embedded skills using an embedded environment. Unlike most Linux courses, that use PC’s as the target, this course uses Power PC target hardware – a true embedded environment.

Overview:

Linux has a world wide support database, and probably the largest supported device driver base of any operating system. However, for the Linux novice the choices can be bewildering and confusing (e.g. what is the difference between Red Hat Linux and Blue Cat Linux?).

This course is unique in a number of areas. First it is not aimed at Linux gurus; it is aimed at existing developers using traditional RTOS’s in embedded environments. Second, it proposes that Linux is not suitable for all applications. It is as important to understand what Linux cannot do as much as what it can do. Third, and most significant, it isn’t based on traditional x86 PC hardware. This course uses a true embedded environment (a Power PC card) that does not have a floppy or hard disk. An environment most embedded engineers are familiar with.

This course demonstrates, through extensive hands-on, how to build small, fast applications with embedded Linux. Note, it does not promote any vendors tools.

Course Objectives:

• To get real-world exposure to embedded Linux

• To develop an application to run on an embedded Linux system

• To understand what is required to set-up a Linux cross development environment

• To understand the different approaches to making Linux “realtime” Delegates will learn:

• How to configure a standard Linux kernel for use in a cross development system.

• The steps to write, compile, download and debug an embedded Linux application with real hardware.

• How threaded applications fit into Linux.

Who Should Attend:

Software engineers who are developing applications for embedded or real-time Linux. Engineers wishing to assess the suitability of Linux for their next application.

Pre-Requisites:

• Good ‘C’ programming skills

• General knowledge of an RTOS or embedded operating systems

• Knowledge of Linux or Unix will be useful, but not essential

• Be able to use basic Linux/Unix commands (e.g. ls, cat, vi).

Duration:

Five days.

Course Materials:

Student workbook.

Course Workshop:

The course presents embedded and real-time concepts applied to Linux, using a Power PC development board as the target (TQ PPC starter kit). The host development system will be a standard PC running RedHat. We use the target as an example of a simple embedded system which can control hardware via a simple digital I/O interface.

Lab sessions follow a logical sequence, and result in “the world’s first Linux-powered web-controlled washing machine”. A distinction is made between the development host and the target throughout.

Course Outline

Introduction

- An overview of Linux’s strengths and weaknesses

Setting up the development

environment

- What tools are needed, where to get them and how to install them.

Introduction to the Linux kernel

- What is in the kernel?

- Configuring and booting the kernel.

The root file system

- Root file directories, /bin /etc/lib etc.

More about Linux file systems

- Types of file system: Disk, RAM, Flash, Network

Some debug techniques

- Syslog and strace. GDB and DDD

TCP/IP Networking

- Network configuration and web servers

Device control from user space

- Accessing hardware directly

Multi processing on Linux and Inter Process Communication

- Linux process model and IPCs

Multithreading using pThreads

- Threads vs. Processes and pThreads

Linux and Real-Time

- Standard kernel problems and patches

Real-time sub kernels

- RTLinux and RTAI

RTAI

- Using RTAI

Conclusion

EL-504

Developing Linux Device Drivers

Length: 5 days

Course Detail

Course Description:

Linux is now being designed into everything from super computers down to mobile phones. Fortunately, to deploy Linux in your design very few applications require the knowledge to re-program the actual Linux kernel. However, most applications do require some customisation for the specific hardware devices in their design. This course caters for the many engineers needing only the skills to design and test a Linux device driver.

Course Objectives:

• To provide an overview of the Linux kernel internal mechanisms available to a driver developer

• To become a proficient Linux driver developer for character, block and network devices

• To recognise resource eating inefficient drivers, and tailor to minimise RT latency.

Delegates will learn:

• The kernel module development cycle: edit, compile,

load, test and unload kernel module drivers

• To write, compile and link simple application programs to test drivers

• To build a kernel with a new built-in driver

• To obtain a high-level appreciation of the relationship between GNU, Open Source, GPL, and LGPL.

Pre-Requisites:

• Good C

• Knowledge of embedded micros, interrupts etc.

• Some Linux or any form of Unix familiarity would be helpful

This course would be an excellent compliment to the Feabhas EL-503: Developing for Embedded Linux. This is not mandatory though.

Who Should Attend:

Engineers interested in interfacing to custom hardware on a Linux platform. Linux application programmers interested in accessing Linux at a lower level.

Duration:

Five days.

Course Material:

• A delegate handbook containing all the theory presentation slides and laboratory exercises

• CD containing all kernels and utilities for the laboratory exercises.

Related courses:

EL-503: Developing for Embedded Linux.

Laboratory Exercises:

• Drivers are written or customized to explore all theoretical material using a PowerPC embedded target running Linux

• For driver debugging, exercises include using conventional printk and oops methods, as well as advanced sessions using the GNU debugger, gdb. The workshops are completed by adding a driver “permanently” to a kernel image.

Course Outline

The GNU/Linux kernel

- A high level overview of the kernel and

terminology including: virtual memory;

user/kernel space; processes; threads

- How to set up a cross-development

environment

- How to configure, cross-compile and

boot Linux on the PPC target

- Real-time behaviour and its impact on

device driver implementation

The kernel programming environment

- Modules: Adding driver modules to the

kernel, module utilities, symbols,

parameters, revisions and compatibility

- /proc: Talking directly to the kernel

- Debugging tools including: gdb, kgdb,

strace, printk, ksyms, lsmod and logging daemons

The device driver interface

- Device nodes, major-minor numbers,

dynamically allocated driver numbers

- Registering character drivers, device
operations: open, release, read and
write
- Transferring data between user and
kernel space
- Blocking on wait queues, re-entrancy
issues
- synchronisation using semaphores and
spinlocks

Kernel mechanisms for the driver

- Kernel memory allocation, virtualphysical, paging, mapping, I/O Ports
- Interrupts: Registering and Handling, Sharing, Disabling, SMP, Bottom half, tasklets, workqueus
- Timing: sources of time, short/long
delays, current time, jiffies, kernel
timers, sleeping
- Poll/select; asynchronous notification
- ioctl

Advanced Kernel Issues

- Block and network drivers
- Peripheral Buses: PCI