Embedded/Real-Time Linux Dev.

Course Description

This course readies software engineers to build or support embedded Linux based products.

Overview

This course provides substantial practice with the key steps in developing an embedded Linux product. The course shows attendees how to configure a small Linux kernel, develop code within the kernel, such as for new system functionality or device drivers, and how to measure and obtain real-time performance with Linux.

The course begins with a complete, simple, project that walks the attendees through the entire process of creating a special version of the Linux kernel, creating a root file system, including just the libraries that are needed, and constructing a custom boot sequence.

Attendees will spend approximately 50 percent of the class time actually gaining hands-on experience with these topics.

Course Objectives

- To provide an understanding of the essentials of embedded and real-time Linux.
- To give you practical experience in developing an embedded Linux system.

Attendees will learn:

- Key concepts and software for embedded Linux.
- Key concepts and software for real-time Linux.
- How to configure a Linux kernel.
- How to measure real-time performance in Linux.
- Fundamentals of embedded software for Linux

Who Should Attend:

The course is designed for real-time or embedded engineers who are new to real-time or embedded Linux. Attendees should have experience with C and be able to perform basic Unix commands.

Duration

Four and one-half days when open enrollment, four days on-site.

Course Materials

The workshop materials include a comprehensive student workbook . The workbook contains all of the slides used in the course as well as hands-on lab exercises.

This course may optionally be taught with the use of a representative embedded Linux device.

Course Workshop:

The workshop makes use of standard PC’s with a desktop Linux distribution for development. The PC will be used as an example target for both real-time and embedded. Alternative platforms such as those with ARM or PPC CPUs will be used as cross targets. Since Linux has been effectively ported to many architectures, the principles taught in the workshop are appropriate for a wide range of target platforms.

Embedded and Real-Time Linux Development Outline

1. Embedded Linux Development
- 1.1. Objectives and format
- 1.2. What/Why/How/Who/Where of embedded Linux

2. Overview Of Project
- 2.1. Building an mp3 playing, web browser controlled, appliance
- 2.2. Configuring a Linux kernel

3. Building A Root File System
- 3.1. What directories are required?
- 3.2. Making busybox
- 3.3. Configuring the boot sequence
- 3.4. Configuring networking

4. Building A System Image
- 4.1. Device drivers
- 4.2. Inserting drivers
- 4.3. Stacked drivers
- 4.4. Libraries

5. Applications
- 5.1. running an embedded web server
- 5.2. mp3 software

6. Making A Boot Image
- 6.1. Putting the pieces together
- 6.2. Creating A Filesystem image
- 6.3. Bootloaders
- 6.4. U-Boot
- 6.5. GRUB
- 6.6. RedBoot

7. File Systems
- 7.1. Flash Devices
- 7.2. Read-Only File Systems
- 7.3. CRAMFS
- 7.4. Journaling File Systems
- 7.5. Benchmarking File Systems

8. Programming with GNU tools
- 8.1. gcc
- 8.2. optimization
- 8.3. linker
- 8.4. debugging with gdb

9. Cross development
- 9.1. Cross compilation
- 9.2. Libraries and tool chains
- 9.3. How to configure the kernel for cross-compiling
- 9.4. Building the kernel and modules

10. Tools
- 10.1. Tracing
- 10.2. Finding Memory Errors
- 10.3. Profiling

11. Kernel And System Programming
- 11.1. Writing a system call
- 11.2. System call basics
- 11.3. Shared memory
- 11.4. Threads
- 11.5. Synchronization, Scheduling
- 11.6. Memory locking

12. Linux and Real Time
- 12.1. What is real time?
- 12.2. a real-time time line
- 12.3. user space vs. kernel space
- 12.4. issues
- 12.5. latencies
- 12.6. low latency patch
- 12.7. linear scheduling
- 12.8. Non-preemptive kernels
- 12.9. latency test tool

13. Preemption
- 13.1. Preemptibility
- 13.2. Low Latency
- 13.3. Preemptible Kernels
- 13.4. Comparing Preemptible Solutions
- 13.5. RT-Preempt
- 13.6. Priority inheritance
- 13.7. Schedulable interrupt threads