Starting Embedded Linux Application Development on ARM based processors

Summary

This course introduces embedded Linux development to existing embedded system-on-chip developers. It focuses on ARM based processors and the ARM DS-5 IDE. Little or no experience of embedded Linux or Linux application development is assumed. The course takes advantage of the DS-5 package to help speed up the learning process and remove impediments, whilst focusing on the requirements of embedded applications (such as low level hardware access & TCP/IP socket communication).

The Challenges of Embedded Linux Application Development

For the embedded software engineer, starting embedded Linux application development can present a major hurdle. Conceptually, things are very different to development using an RTOS or bare metal programming… Linux shows its mainframe heritage. To further complicate matters, embedded Linux application development normally requires development tools that run under Linux. The learner therefore needs to first install and learn the basics of desktop Linux. Once this hurdle has been surmounted, there are then various choices for open source development tools: both command line and IDE-based.

Although this wide choice is very powerful, it represents a fierce challenge as one attempts to find a learning path. The purpose of this Course is to provide a unified, coordinated path for embedded developers starting out in embedded Linux programming. There are actually very few books which focus on embedded Linux *programming*. Once you have a working Linux platform the newbie developer is faced with the question 'What do I do Next?' This course is designed to answer that question.

A One Step Development System

The course uses ARM's DS-5 development package. DS-5 provides a full development environment for embedded Linux as a one step installation running under Windows. DS-5 comprises a full IDE, cross compiler, debugger,profiler, hardware simulator and example applications. It has dedicated support for Standard Application Specific Devices based on ARM processors (LPC300, SpEAR, iMX, OMAP, Sitara etc) and provides a unified end to end development toolchain. Skills learnt using DS-5 are also applicable to other open source tools.

hitex DEVELOPMENT TOOLS

Contents Summary

An Introduction to Linux and DS-5

- A brief history of Linux and embedded Linux. How Linux started as a mainframe OS and is now happily working inside a watch.
- Advantages of Linux for the embedded developer.
- A summary of the complexities of getting started in embedded Linux development: the lack of a definite learning path, the steep learning curve, the need to use desktop Linux, multiple tool choices, a focus on the command-line, etc.
- A summary of how DS-5 can aid the learning process: a single Windows install of all tools required, a cutting edge IDE that merges open-source tools with ARM's own components.

First Steps with DS-5

- Versions of DS-5.
- Running the Gnometrix example on the supplied ARM simulator
- Creating a simple 'Hello World' application.
- A first look at the DS-5 debugger

Target Hardware

- An introduction to ARM target hardware.
- How to connect and set up the hardware.
- How to create a Linux OS on a Memory card from a downloaded image

A First Application Hello GPIO

At this point we have DS-5 up and running and have looked at the examples. We also have target hardware up and running. We now need to write our very first embedded application.

- Hello GPIO will flash the User LED on the course hardware.
  It accomplishes this using the Kernel GPIO driver and sysfs filesystem.

Introduction to the Linux Command Line

One of the major differences between Linux and a standard embedded OS (such as an embedded RTOS) is the omnipresent command line 'shell'. The shell changes the way one thinks about embedded software: no longer is it simply a binary which transfers over to a target and then executes. Instead, it can be a series of programs present within a filesystem. These programs can be started, stopped, configured and monitored from the command line. The command line therefore provides an excellent tool for experimentation, diagnostics and debugging during development, testing and even whilst in the field.

- This section explains how to use the command line to perform basic tasks such as:
  - Navigating directories.
  - Listing files
  - Executing programs
  - Editing files
- The following topics are briefly covered:

- Busybox. Most embedded distributions use this tool for the command line shell, editor and command line toolset.
- Basic command line features, such as stdout and stdin.
- A simple crib for the vi editor: how to move around, how to perform basic editing, how to save and exit.

• A look at the basic structure of the Linux filesystem, as seen whilst walking the directories from the command line.
• A description of security issues and user permissions in Linux.
Although embedded Linux tends to run internally as root user, an understanding of permissions is important as permissions problems are a frequent stumbling block when starting Linux development. The following will be summarised:

- Users and the root user.
- Read, write and execute permissions for file owners, group members and others.
- What directory permissions mean.
- A brief description of special permissions, such as how to give a program root execute permissions.
- How to change permissions with chmod and chgrp command line tools. How to manage permissions within DS-5
- Typical permissions issues that embedded developers often face.

Device Drivers in Linux

• The difference between the kernel and userland. How userland application space is normally protected and direct I/O performed by drivers inside the kernel.
• Device drivers in the kernel: their basic structure.
• How to communicate with a kernel device driver from the command line.
• How to communicate with a kernel device driver in C. Opening files for reading and writing, sending commands and changing settings.
• Using libraries to communicate with I/O instead of communicating directly with the kernel drivers.
• Common device drivers:

- GPIO driver - access I/O directly.
- SPI and I2C drivers
- Real Time Clock drivers
- Memory Drivers (mention different filesystems: ext2, ext3, flash filesystem)
- LCD framebuffer driver

• Finding out what I/O has been sensed by the kernel during bootup and which drivers are available.

Processes and Interprocess Communications

• The concept of a process: how to do more than one thing at a time. How to monitor processes from the command line.
• Creating processes in C.
• Interprocess communications: passing messages between processes via shared memory, signals, pipes and semaphores.
• A practical example, using a couple of LED and buttons. Separate tasks for LED and buttons; communication of button state between tasks.

Debugging Code

• Debugging code using the DS-5 debugger. An introduction to the various elements of the DS-5 debugger.

Profiling and Performance Tuning

• Overview of the DS-5 profiler.
• Working with the DS-5 profiler to understand the execution of a block of code then tune it for performance.

Sockets and TCP/IP

Embedded systems are increasingly using socket-based communications to communicate with the outside world. A primary reason for choosing embedded Linux is the free, mature, built-in TCP/IP communications stack and tools.

• Fundamentals of how sockets are handled in C, to send and receive TCP/IP requests.
• Example of how to send HTTP GET and POST requests to interrogate devices and send information to central servers.
• Embedded Linux web servers, from tiny servers such as thttpd and micro_httpd to Apache. Handling HTTP requests in C, using CGI.
• Creating an embedded web browser for an embedded system that allows users to configure an embedded application.
• A practical example using TCP/IP to retrieve temperature values from a number of small embedded devices, then sending the aggregated data to a remote server on the internet.

Where Do I Go From Here?

• A summary of sources of embedded information.
• Consultancy services

Course Details

Location

The course is run at the Offices of Hitex UK on Warwick University Science Park, CV4 7EZ.

On site courses are also available on request.

Class size

To ensure a good Instructor to student ratio the maximum class size is six delegates.

Duration

This is a one day course and runs approximately 9.30 – 4.30. Refreshments and lunch are provided.

Booking

Please book via our webshop at

http://www.hitex.co.uk/fileadmin/uk-files/shop/contents/en-uk/d47.html

Or contact Louise Taylor ltaylor@hitex.co.uk