Invotronics

Embedding Ideas into Reality

Embedded Linux System Development

5 Day Session

Overview

Title	**Embedded Linux System Development Training**
Overview	Boot loaders. Kernel (cross) compiling and booting. Block and flash file systems. C library and cross-compiling tool-chains. Lightweight building blocks for embedded systems. Embedded system development tools. Embedded application development and debugging. Implementing real-time requirements in embedded Linux systems. Techniques to optimize system size, RAM, power, performance and cost. Practical labs with ARM boards.
Duration	5 days - 40 hours (8 hours per day) 40% of presentations and 60% of practical labs.
Trainer	One of the engineers listed on http://invo-tronics.com/training/trainers/
Language	Oral lectures: English Materials: English
Audience	People developing devices using the Linux kernel. People supporting embedded Linux system developers.
Prerequisites	Knowledge and practice of Unix or GNU/Linux commands People lacking experience on this topic should get trained by themselves with our freely available on-line slides (http://invo-tronics.com/docs/command-line/)
Required Equipment	Video projector. PC computers with at least 2 GB of RAM, and Ubuntu Linux installed in a free partition of at least 20 GB. Using Linux in a virtual machine is not supported, because of issues connecting to real hardware. We need Ubuntu Desktop 12.04 (32 or 64 bit, Xubuntu and Kubuntu variants are fine). We don't support other distributions, because we can't test all possible package versions. Connection to the Internet (direct or through the proxy). PC computers with valuable data must be backed up before being used in our sessions. Some people have already made mistakes during our sessions and damaged work data.
Materials	Print and electronic copies of presentations and labs. Electronic copy of lab files.

Copyright © 2014 Invotronics, http://invo-tronics.com/training/, training@invo-tronics.com

Invotronics

Embedding Ideas into Reality

Training – Embedded Linux System Development

See our training materials on http://invo-tronics.com/doc/training/embedded-linux
This way, you can check by yourself that they correspond to your needs.

Hardware

Using IGEPv2 boards from ISEE in most practical labs	
DM3730 (OMAP3) CPU from Texas Instruments 512 MB RAM, 512 MB flash 1 USB 2.0 host 1 USB device HDMI / DVI-D video output, audio I/O 100 Mbit Ethernet port, Wifi, Bluetooth Expansion port, JTAG port, etc.	

Day – 1 Morning

Lecture – Introduction to embedded Linux	Lecture – Embedded Linux development environment
Advantages of Linux versus traditional embedded operating systems. Reasons for choosing Linux. Global picture: understanding the general architecture of an embedded Linux system. Overview of the major components in a typical system. The rest of the course will study each of these components in detail.	Operating system and tools to use on the development workstation for embedded Linux development. Desktop Linux usage tips.
Lecture - Cross-compiling tool-chain and C library	**Lab – Cross compiling tool-chain**
What's inside a cross-compiling tool-chain Choosing the target C library What's inside the C library Ready to use cross-compiling tool-chains Building a cross-compiling tool-chain with automated tools.	Configuring Crosstool-NG Executing it to build a custom uClibc tool-chain.

Day – 1 Afternoon

Lecture – Boot-loaders	Lab – Boot-loader and U-boot
Available boot-loaders Boot-loader features Installing a boot-loader Detailed study of U-Boot	Set up serial communication with the board. Configure, compile and install the first-stage boot-loader and U-Boot on the IGEPv2 board. Become familiar with U-Boot environment and commands. Set up TFTP communication with the board. Use TFTP U-Boot commands.
Lecture – Linux kernel	Lab - Kernel sources

Copyright © 2014 Invotronics, http://invo-tronics.com/training/, training@invo-tronics.com

Role and general architecture of the Linux kernel Features available in the Linux kernel, with a focus on features useful for embedded systems Kernel user interface Getting the sources Understanding Linux kernel versions. Using the patch command	Downloading kernel sources Apply kernel patches

Day – 2 Morning

Lecture – Configuring and compiling a Linux kernel	**Lecture - Kernel cross-compiling**
Kernel configuration. Useful settings for embedded systems. Native compiling. Generated files. Using kernel modules	Kernel cross-compiling setup. Using ready-made configuration files for specific architectures and boards. Cross-compiling Linux.
Lab - Kernel cross-compiling and booting	
Using the IGEPv2 ARM board Configuring the Linux kernel and cross compiling it for the ARM board. Downloading your kernel on the board through U-boot's tftp client. Booting your kernel from RAM. Copying the kernel to flash and booting it from this location. Storing boot parameters in flash and automating kernel booting from flash.	

Day – 2 Afternoon

Lecture – Root file system in Linux	**Lecture - Busy Box**
File systems in Linux. Role and organization of the root file system. Location of the root file system: on storage, in memory, from the network. Device files, virtual file systems. Contents of a typical root file system.	Detailed overview. Detailed features. Configuration, compiling and deploying.
Lab – Tiny root file system built from scratch with Busy Box	
Now build a basic root file system from scratch for your ARM system Setting up a kernel to boot your system on a workstation directory exported by NFS Passing kernel command line parameters to boot on NFS Creating the full root file system from scratch. Populating it with Busy Box based utilities. Creating device files and booting the virtual system. System start-up using Busy Box /sbin/init Using the Busy Box http server. Controlling the target from a web browser on the PC host. Setting up shared libraries on the target and developing a sample application.	

Day – 3 Morning

Lab – Tiny root file system built from scratch with Busy Box
Continued from the previous afternoon.

Copyright © 2014 Invotronics, http://invo-tronics.com/training/, training@invo-tronics.com

Invotronics

Embedding Ideas into Reality

Lecture - Block file systems	Lab - Block file systems
File systems for block devices. Usefulness of journaled file systems. Read-only block file systems. RAM file systems. How to create each of these file systems. Suggestions for embedded systems.	Using the IGEP ARM board Creating partitions on your block storage Booting a system with a mix of file systems: SquashFS for applications, ext3 for configuration and user data, and tmpfs for temporary system files.

Day – 3 Afternoon

Lecture - Flash file systems	Lab – Flash file systems
The Memory Technology Devices (MTD) file system. File systems for MTD storage: JFFS2, Yaffs2, UBIFS. Kernel configuration options MTD storage partitions. Mounting MTD file system images.	Using the IGEPv2 ARM board Creating partitions in your internal flash storage. Formatting the main partition with JFFS2 in read only mode. Using JFFS2 for system data.
Lecture – Leveraging existing open-source components in your system	
Reasons for leveraging existing components. Find existing free and open source software components. Choosing the components. The different free software licenses and their requirements. Overview of well-known typical components used in embedded systems : graphical libraries and systems (framebuffer, DirectFB, Gtk, Qt, etc.), system utilities, network libraries and utilities, multimedia libraries, etc. Example of a typical consumer electronic product leveraging many open-source components. System building: integration of the components.	

Day – 4 Morning

Lecture – Cross-compiling applications and libraries	Lab – Cross-compiling applications and libraries
Configuring, cross-compiling and installing applications and libraries. Details about the build system used in most open source components. Overview of the common issues found when using these components.	Building a system with a graphical system based on DirectFB, running in Qemu. Manual compilation and installation of several free software packages. Learning about common techniques and issues.

Day – 4 Afternoon

Lecture - Embedded system building tools	Lab - System build with Buildroot
Review of existing system building tools. Buildroot example.	Building a system with a graphical system based on DirectFB, running in Qemu. Using Buildroot to rebuild the same system as in the previous lab. Seeing how easier it gets. Add a package to Buildroot.

Copyright © 2014 Invotronics, http://invo-tronics.com/training/, training@invo-tronics.com

Invotronics

Embedding Ideas into Reality

Day – 5 Morning

Lecture - Application development and debugging	Lab – Application development and debugging
Programming languages and libraries available. Overview of the C library features for application development. Build system for your application, how to use existing libraries in your application. Source browsers and Integrated Development Environments (IDEs). Debuggers. Debugging remote applications with gdb and gdbserver. Post-mortem debugging with core files. Code checkers, memory checkers, profilers. Developing on Windows.	In Qemu and on the IGEP ARM board. Develop and compile an application relying on the DirectFB library. Using strace, ltrace and gdbserver to debug a crappy application on the remote system. Do post-mortem analysis of a crashed application.

Day – 5 Afternoon

Lecture - Linux and real-time	Lab - Linux latency tests
Very useful for many kinds of devices, industrial or multimedia systems. Understanding the sources of latency in standard Linux. Soft real-time solutions for Linux: improvements brought by Linux 2.6. Understanding and using the latest RT preempt patches for mainstream Linux. Real-time kernel debugging. Measuring and analyzing latency. Xenomai, a hard real-time solution for Linux: features, concepts, implementation and examples.	Tests performed on the IGEPv2 ARM board Latency tests on standard Linux. Setting up Xenomai. Latency tests with Xenomai.

Copyright © 2014 Invotronics, http://invo-tronics.com/training/, training@invo-tronics.com