

Embedded Linux Training

Title	Embedded Linux Training
Overview	3 days: Linux kernel for embedded systems. Device driver development. Caution: focused on driver development, not on core kernel details. Practical labs with virtual x86 and ARM systems and with Technologic Systems' TS-5500 boards.
Duration	3 days. 2/3 of presentations and 1/3 of practical labs.
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 self-trained at (http://free-electrons.com/training/intro_unix_linux/) Knowledge and practice of C programming Familiarity with written English
Training Facilities	1 PC on each desk for each participant Technologic TS-5500 Single Board Computer Internet Connection
Materials	Print and electronic copy of presentations and labs. Electronic copy of lab files.
Date	Nov. 14, 2007 - Nov 16, 2007
Time	9.00am - 6.00pm
Venue	Meeting Room 5, Vistana Hotel, Penang
Training Fees	RM3,000.00 per participant (Training Fees include course materials, lab files, lunch & Coffee/Tea)
Early Bird Discount	RM500 discount for registration and full payment made before October 1, 2007
Cancellation	50% refund where cancellation in writing before November 1, 2007
Replacement	Replacement participant within the same organization is allowed and MUST be made in writing 2 days before training date
Enrollment	Name: _____ Job Title: _____ Organization: _____ Tel No.: _____ Address: _____ Email: _____
Registration	Please send completed registration form with a purchase order and payment cheque payable to Netshoppe Sdn Bhd to: Netshoppe Sdn Bhd 79-B, Jalan SS21/37 Damansara Utama 47400 Petaling Jaya Selangor. Tel. 603-7726-2840 Fax. 603-7726-2850

Training contents

Introduction

- Legal questions about Free and Open Source Software
- System administration basics

Linux kernel and driver development

- Linux kernel introduction
- Kernel sources
- Compiling
- Booting
- Cross-compiling
- Basic driver development
- Writing modules
- Memory management
- I/O memory
- Character drivers
- Driver development techniques
- Interrupts
- Udev and hotplugging
- Choosing filesystems
- Advice and resources

Embedded Linux Training

Day 1 - Morning

Lecture - Legal questions about Free and Open Source Software	Lecture - System administration basics
Free Software and Open Source licenses Constraints for embedded system developers Potential issues with Free Software licenses Software patents Legal support resources	... Very useful in embedded systems! Misc: file ownership, shutting down... Setting up networking Filesystems: creating and mounting

Practical lab - Lab environment setup	Lecture - Linux kernel introduction
Using the KernelKit GNU/Linux live CD (http://kernelkit.org). If needed, creating free space by shrinking Windows partitions on the PC hard disk. Creating a new partition on the hard disk with fdisk and formatting it for Linux. To speed up practical labs, installing KernelKit on the hard disk.	System and kernel overview Linux features Kernel code specificities Kernel subsystems Processes and scheduling Supported hardware architectures History and versioning scheme. Understanding the development process. Specific legal issues. How major companies handle these requirements. Kernel user interface

Day 1 - Afternoon

Lecture - Kernel sources	Practical lab - Kernel Sources
Getting the sources Checking their authenticity Using the patch command Structure of source files Kernel source code browsers	Getting kernel sources, checking their authenticity. Applying kernel patches. Exploring sources in search for files, function headers or other kinds of information...

Lecture - Compiling
Kernel configuration. Useful settings for embedded systems. Compiling. Generated files

Lecture - Booting	Practical Lab - Compiling and booting
<p>Linux system booting overview. The bootloader's job. Review of Linux bootloaders. U-boot and GRUB details. Linux kernel booting. Advantages of initramfs over initrd. Booting parameters. NFS boot example. System startup</p>	<p>Using the Technologic Systems boards Configuring, compiling and booting Linux. Installing the GRUB bootloader on the board. Modifying a root filesystem image by adding entries to the /dev/ directory.</p>

Day 2 - Morning

Lecture - Cross-compiling	Practical lab - Cross-compiling
<p>Kernel cross-compiling setup. Using ready-made configuration files for specific architectures and boards. Cross-compiling</p>	<p>Setting up a cross-compiling environment for the ARM & AMD instruction set. Configuring Linux kernel and cross-compiling it for virtual ARM system & TS-5500. Booting the compiled Linux kernel on the virtual ARM system & TS-5500.</p>

Lecture - Basic driver development
<p>Linux device drivers A simple module Programming constraints Loading, unloading modules Module parameters Module dependencies Adding sources to the kernel tree</p>

Day 2 - Afternoon

Practical lab - Writing modules	Lecture - Memory management
<p>Write a kernel module with several capabilities, including module parameters and a /proc output interface. Access any kernel internals from your module. Setup the environment to compile it</p>	<p>Linux: memory management - Physical and virtual (kernel and user) address spaces. Linux memory management implementation. Allocating with kmalloc(). Allocating by pages. Using lookaside caches and memory pools. Allocating with vmalloc().</p>

Lecture - I/O memory	Lecture - Character drivers
I/O register and memory range registration. I/O register and memory access. Read / write memory barriers.	Device numbers Getting free device numbers File operations Character driver registration

Day 3 - Morning

Practical lab - Character drivers
Writing a module for your workstation kernel, without having to have its sources Writing a simple character driver Having the kernel allocate a free major number for you, and create a device file accordingly. Writing simple file_operations functions for a device, including ioctl controls. Using the kmalloc and kfree utilities Copying data from user memory space to kernel memory space and vice-versa. Practicing kernel standard error codes

Day 3 - Afternoon

Lecture - Driver development techniques	Practical lab - Tracing kernel code
Kernel debugging techniques Managing concurrent access to resources: mutexes, spinlocks. Atomic operations	Using SystemTap to trace selected kernel functions: access parameters, follow pointers, collect statistics...

Lecture - Interrupts	Lecture - Choosing filesystems
Waiting for the availability of resources Interrupt handler registration Interrupt handlers Scheduling deferred work.	Filesystems for block devices. Usefulness of journaled filesystems. Read-only block filesystems. RAM filesystems. Filesystems for Memory Technology Devices (MTD), such as flash chips. Suggestions for embedded systems.

Lecture - Advice and resources	Lecture - Udev and hotplugging
System security guidelines Making portable drivers (endianness independence, etc.) Getting help and contributions Bug report and patch submission to Linux developers. References: websites, books and international conferences	If time left New device model, sysfs. Udev: handling hardware events from userspace: creating and removing device files, identifying drivers, notifying programs and users.

About the trainer - Michael Opdenacker

Born in 1972. French and Belgian. Married.

Michael has a 15 years experience using Unix and GNU / Linux and promoting Free Software within the companies he worked for.

Michael's best resume is Google. See by yourself!

<http://google.com/search?q=michael+opdenacker>

▪ **Embedded Linux experience**

Michael has been working on embedded Linux since 2003:

- Creating the Free Electrons company, to support individuals and organizations in using Linux and Free Software in embedded systems.
- Creating training materials and presentations (see <http://free-electrons.com/training>). Performing training sessions for major embedded platform suppliers and embedded system makers.
- Doing consulting and support work on embedded Linux systems and developing demos.
- Participating to handhelds.org's projects to port Linux on PDAs, writing kernel drivers, documentation and creating root filesystems. First Linux boot on the HP iPAQ h2200 device.
- Working on system startup time. Public contributions so far: embedded Linux optimization presentation (<http://free-electrons.com/articles/optimizations/>), Busybox [readahead applet](#).
- Creating a live CD (<http://kernelkit.org>) for embedded Linux development and training purposes.
- Regularly making public presentations in national and international technical conferences.
- Participating to international technical conferences and [sharing videos with the community](#).

▪ **Free Software development**

Since 2000, active contributor to the Free Software community. In particular:

- Many contributions through Free Electrons: <http://free-electrons.com/community/contributions>. Created resources attract approximately 1000 daily visitors (monthly average) to our website.
- 2003: Author of [Minido](#), a simple yet generic task manager written in GTK2.
- 2001-2003: Maintainer of the GNU Typist project. Took care of makefiles (autoconf / automake), translations (gettext), documentation (makeinfo) and releases.

▪ **Microelectronics**

Michael is also very familiar with the microelectronics industry:

- 1995-2000: worked for [STMicroelectronics](#), Central R&D, Design Automation Department, Grenoble, France. Developed I/O and memory cell generators, as well as cell library validation tools. Trained and supported users in France, Italy and India.
- 2000-2003: worked for [Texas Instruments](#), Wireless Business Unit, Nice, France. Took care of front-line support on chip physical design tools (Synopsys, Magma, internal tools). Migrated design flows and compute farm to Linux servers.

▪ **Education**

- 1991-1994: [ENSIMAG](#) Engineering School, Grenoble, France
Engineer diploma in Computer Science and Applied Mathematics.
- 1993-1994: [University of Grenoble](#), France
DEA postgraduate diploma in Artificial Intelligence.

Curriculum Advisor – Assoc. Prof. Dr. R. Badlishah Ahmad

Assoc Prof Dr. R. Badlishah Ahmad is Dean, School of Computer & Communication Engineering at University Malaysia Perlis, UNIMAP. He is a strong advocate and an emerging voice of Linux in embedded systems in the computer and communication engineering space.