Embedded Linux

*Device driver development*

Sébastien Bilavarn
Module organisation

- Linux Lectures (3x2h)
- Labs (5x3h)
  - Xilinx Zynq-7000 / Linux (ZedBoard)
  - IOCTL, framebuffer (Exynos/Ubuntu)
- Evaluation
  - Lab: evaluated reports (mean value of 5 lab reports)
  - Exam: final examination (2h)
  - Final score = \((2\times\text{Exam}) + \text{Lab}\) / 3

- Online resources
  - [http://users.polytech.unice.fr/~bilavarn/](http://users.polytech.unice.fr/~bilavarn/)
    - login: gse5 / pwd: kintex7
  - Bibliography, lab resources
Outline

- Ch1 – Introduction to Linux
- Ch2 – Linux kernel overview
- Ch3 – Linux for Embedded Systems
- Ch4 – Embedded Linux distributions
- Ch5 – Case study: Xilinx PowerPC Linux
- Ch5 bis – Case study: Xilinx Zynq-7000 Linux
- Ch6 – Device driver development
Introduction to Linux

- History
- Overview
- Linux basics
- Filesystem
- Administration
- Commands
At the beginning of computer history, size was a problem. Software as well: software was always customized to serve a specific purpose, every computer had a different operating system.

In 1969, a team of developers in the AT&T Bell Labs laboratories started working on a solution for the software problem. They developed a new operating system, which was simple and elegant. Written in C language instead of assembly code. Able to recycle code.

The Bell Labs developers named their project "UNIX".
History

UNIX

- Before Unix, programs were specifically developed for one system.
- In UNIX, only a small piece of code is specific to the system: the *kernel*.
  - The kernel is the only piece of code that needs to be adapted for every specific system and forms the base of the UNIX system.
  - The operating system and all other functions were built around the kernel and written in a higher programming language, C.
  - C language and UNIX were developed almost at the same time. UNIX is closely related to C language. UNIX was rewritten in C.
  - Using this new technique, it was much easier to develop an operating system that could run on many different types of hardware.
UNIX

- UNIX was initially found only in very large environments with mainframes and minicomputers. You had to work at a university, for the government or for large financial corporations in order to get your hands on a UNIX system.

- By the end of the 80's, many people had home computers. There were several versions of UNIX available for the PC architecture, but none of them were truly free and they were very slow.

- At the beginning of the 90s home PCs were finally powerful enough to run UNIX.

- Linus Torvalds, a young student studying computer science at the university of Helsinki, thought it would be a good idea to have some sort of freely available academic version of UNIX, and started to develop in 1991.
Linux story, short version

- Free system completely compliant with the original UNIX.
- The full system is based on GNU tools: C libraries, gcc, binutils, fileutils, make, emacs…
- The full system is called « GNU/Linux »
- Shared very early as Free software (GPL License), Linux attracted a growing number of contributors and users since 1991, exceeding any other operating systems (not only Unix)

http://www.youtube.com/watch?v=5ocq6_3-nEw
Introduction to Linux

- History
- Overview
- Linux basics
- Filesystem
- Administration
- Commands
Why Linux?

- Quality and reliability of code
  - Modularity and structural design
  - Debug facilities
  - Easy to expand
  - Configurable
  - Error recovery (diagnostic messages)
  - Duration (contribution of a large open source community)

- Code availability
  - The majority of code is available with no restriction (GNU GPL)
  - Problems are fixed quickly and for free by the open source community

- Hardware support
  - A variety of platforms supported (x86, ARM, PowerPC, MIPS, 68000, OMAP)
  - Driver adaptation is made easier
Why Linux?

- Support of communication protocols and applications
- Available tools
- Linux developers community
- License
  - High degree of freedom
- Free
  - Sources available for free: no royalties
  - Vendor independence
- Low cost
- Real time extension
- Multiprocessor support
What is a Linux distribution?

- Strictly speaking, it refers to the *kernel* developed and maintained by Linus Torvald, and distributed via repositories accessible on the web.
  - Official distribution: www.kernel.org
  - General distributions: Debian, Gentoo, Red Hat/Fedora, Ubuntu
Linux distributions

- Differences between Linux distributions
  - Depend on usage, size and price
  - Have in common to provide a set of files and a Linux installation procedure to run the kernel (and other software) on a specific platform.
  - There are several methods for Linux installation (CD-ROM, archive).
Linux versionning

- One stable major branch every 2 or 3 years
  - Identified by an even middle number
  - Examples: 1.0, 2.0, 2.2, 2.4
- A development branch is identified by an odd middle number
  - Provides new functionalities and major changes
  - Examples: 2.1, 2.3, 2.5
- After some time, a development version becomes the new base version for the stable branch
- Minor releases once in a while: 2.2.23, 2.5.12, etc.
Linux versionning

- New 3.X branch
  - From 2003 to 2011, the official kernel versions were named 2.6.x.
    - End of even-odd alternation between stable and unstable
    - Replaced by –rc (release candidates)
  - Linux 3.0 was released in July 2011
  - There is no change to the development model, only a change to the numbering scheme
    - Official kernel versions will be named 3.x (3.0, 3.1, 3.2, etc.)
    - Stabilized versions will be named 3.x.y (3.0.2, 3.4.3, etc.)
    - It effectively only removes a digit compared to the previous numbering scheme: “3.” is used instead of ”2.6”.
  - Linux 4.0 released in April 2015 following version 3.19
    - Kernel version used by Skynet for the T-800 Terminator
Linux source code organisation

- Official distribution:  

Linux is C code
Introduction to Linux

- History
- Overview
  - Linux basics
  - Filesystem
  - Administration
- Commands
Linux basics

- Characteristics
  - File system
    - Linux is a file based management system
    - Device control
      - Storage devices are « mounted », in order to have an associated location in the directory tree.
  - Power of command based management
    - Concept of shell interpreter
  - Administration
    - File permissions and access
      - Concepts of user, super user (root), group
      - Permissions to access files (rwx)
    - Advanced task and process control
  - Kernel modularity
    - Additional modules can be loaded / removed at run time (during execution, after boot)
Linux basics

- Linux interface with the hardware
  - Kernel
    - Part of the operating system that interacts with the hardware
  - Shell
    - Part of the operating system that interacts with the user
Linux basics

- Linux implementations with GUI
Introduction to Linux

- History
- Overview
- Linux basics
- Filesystem
- Administration
- Commands
Almost everything in Linux is a file

- Standard files
- Directories
  - Directories are nothing other than files listing other files
- Symbolic links
  - File referring to the name of another file
- Device drivers
  - Accessing devices (read/write) is based on the use of files.
- Pipes (« | »)
  - Used to cascade several programs
  - Ex: `cat *.log | grep error`
- Sockets
  - Inter processus communication
File names

- File names have the following characteristics:
  - Case sensitive
  - No limited length
  - All characters allowed (except « / »)
  - File name extensions do not require interpretation. They just need to be meaningful for users.
- Exemples
  - README .bashrc Windows Buglist
  - index.htm index.html index.html.old
A path is a sequence of directories with a file or a directory at the end, and separated by « / »

- **Relative path**
  - Relative to the current directory
  - Ex: `documents/fun/microsoft_jokes.html`

- **Absolute path**
  - Ex: `/home/bill/bugs/crash9402031614568`

- **Root directory**
  - The root directory is the start of the absolute paths for all files in the file system
  - « / »
GNU/Linux Root Filesystem (1)

- Nothing is imposed by the system. The Root Filesystem can be different from a system to another, even between two GNU/Linux installations.

```
/       Root directory
/bin    Basic system commands
/boot   Files used by the bootloader (Images, initrd and kernel configuration files)
/dev    Device related files
         /dev/hda: first IDE hard drive
/etc    Platform specific configuration files
/home   User directories
/lib    Basic (shared) system libraries
/mnt    Temporary mount point (by root)
         /mnt/usbdisk
```
GNU/Linux Root Filesystem (2)

/opt Specific tools installed by root
Often replaced by /usr/local

/proc Provides system related information
/proc/cpuinfo, /proc/version, ...

/root Root home directory

/sbin Root specific commands

/sys For system and device control
(proc. frequency, device management, etc.)

/tmp Temporary files

/usr Standard user programs, not essential to the system

/var Data used by the system or its servers
/var/spool/mail (incoming mail), /var/spool/lpd (printing jobs), ...
Introduction to Linux

- History
- Overview
- Linux basics
- Filesystem
- Administration
- Commands
Linux administration

- User privileges
  - Linux is a multi-user system
  - Users
    - Identified by a login/password, each user has its own working directory
      (/home/<user>)
    - A user can not access other users or system’s directories
      → safety, this reduces potential system errors
  - Super user (root)
    - The super user has total control on the system

Files and directories permissions

- General permissions:
  - read, write, execute (rwx)
  - Permissions can be modified by chmod
Permission groups

- Permissions are divided in 3 catégories: user, group, other

Example:

- `$ ls -als`
- `-rwxrwxr-x user1 group1`
  - user1 is the owner and has read, write, execute permissions
  - Each user is part of a group of users: user1 is part of group1
  - Members of group1 have read, write, execute permissions
  - Other users have read, write, execute permissions

- Changing the owner of a file/directory: `chown`
- Changing the group of a file/directory: `chgrp`
- Groups of users are described in `/etc/groups`
Process management

- Each program (process) running on the system is given a specific number:
  - Process Identifier (PID)
  - Some of them are system processes (daemons), that run as background processes
    - Example: sshd is a daemon that services incoming SSH connections
  - The `ps` command returns information on all running processes
    - `ps -aux` returns the list of running processes with their PID
    - Ex: 8053 pts/4 00:00:00 mplayer
  - Sometimes, a process ends but can not be erased from system memory
  - It can be removed with the `kill` command
    - `kill -9 8053` removes previous process mplayer

\!

Like the `rm` command, use with great care. System processes could be killed and lead to system crash or unstability.
Introduction to Linux

- History
- Overview
- Linux basics
- Filesystem
- Administration
- Commands
Command shell

The command shell

- A tool to execute user commands
- Called « shell » because it hides implementation details of the operating system.
- Commands are entered in text mode, either in a specific terminal window in a graphical environment, or in a console for a system without GUI (text only display).
- Command results are also displayed in the terminal. No graphical component is required to run a Linux command.
- Command shells can be programmable: they provide all resources required to write complex programs called script shells (variables, conditions, boucles…).
Usual command shells

- Popular and widely used command shells are
  - sh: the Bourne shell (obsolete)
    - Basic shell traditionnally used in Unix systems, developped by Steve Bourne
  - csh: the C shell (obsolete)
    - Shell based on a C-like syntax, widely used in the past
  - tcsh: the TC shell (still used)
    - C shell compatible, with advanced fonctionnalities (automatic completion of command names, terminal command history, etc.)
  - bash: thee Bourne Again shell (mostly used)
    - Enhanced version of sh with many additional functions
Command edition

- Use left / right arrows to move cursor in the command line
- Use [ctrl][a] to reach beginning of line and [ctrl][e] for the end
- Use up / down arrows to access previous commands
- Use tab for automatic completion of a command
- history: displays last commands that can be reused by copy / paste
Shell variables

- Shell variables (bash)
  projdir=/home/marshall/gadgets
  ls -la $projdir; cd $projdir

- Environment variables (bash)
  PATH=$PATH:/opt/eldk/usr/bin:/opt/eldk/bin
  export PATH

  (export makes the redefinition of PATH effective and accessible from any new terminal window)
Standard environment variables

- Standard environment variables are used by many applications
  - HOME
    - Path of the home directory of the current user
  - HOSTNAME
    - Name of local machine
  - PATH
    - Specifies an ordered list of paths where the shell is able to find effective commands
      Ex: /home/acox/bin:/usr/local/bin:/usr/kerberos/bin:/usr/bin:/bin:/usr/X11R6/bin:/bin:/usr/bin
  - SHELL
    - Name of current shell
  - USER
    - Name of current user
Other environment variables

The following environment variables are very useful in cross-development:

- **ARCH**
  - Allows to specify a target architecture for kernel cross-compilation
  - Ex: arm, powerpc

- **CROSS_COMPILE**
  - Prefix used to specify the cross-compiling tool.
  - Ex: arm-eabi-, powerpc-linux-
  - This prefix is concatenated to the gcc command when a Makefile is invoked.
.bashrc

- .bashrc is a shell script executed each time a shell bash is run (or each time a terminal using a shell bash is opened).
- It is thus very convenient to use .bashrc to add specific environment variables
  - Ex: `export PATH=$PATH:/opt/eldk/usr/bin:/opt/eldk/bin`
Basic commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>User manual of a command</td>
</tr>
<tr>
<td></td>
<td><strong>ex:</strong> <code>man cp</code></td>
</tr>
<tr>
<td>ls</td>
<td>Lists the content of a directory</td>
</tr>
<tr>
<td>cd</td>
<td>Change the current directory</td>
</tr>
<tr>
<td>pwd</td>
<td>Displays the full path of the current directory</td>
</tr>
<tr>
<td>cp</td>
<td>To copy files or directories</td>
</tr>
<tr>
<td>rm</td>
<td>To remove files or directories</td>
</tr>
<tr>
<td>mv</td>
<td>To move files or directories</td>
</tr>
<tr>
<td>mkdir</td>
<td>Creates a new directory (make directory)</td>
</tr>
<tr>
<td>rmdir</td>
<td>To remove an empty directory</td>
</tr>
<tr>
<td>cat</td>
<td>Prints the content of the files specified one by one</td>
</tr>
<tr>
<td>more</td>
<td>More details: at the end of each page displayed, the user is asked to press a key to access the next one.</td>
</tr>
<tr>
<td>grep</td>
<td>Prints the lines corresponding to a pattern in a list of files</td>
</tr>
</tbody>
</table>
Pipes

Usage

- Cascade commands
  - Pipelines the output from one command to the input of another

Example

- `ls -al | grep '^d'`
  - Lists only the lines where the first character is the letter d
Symbolic links

Usage

- Defines a kind of shortcut with a file / directory
  - Allows to access the same file / directory from different locations
  - Avoids the duplication of the file concerned, saves disk space
  - There is only one original file, less errors

Example

- `ls -l file2`
  
  ```
  lrwxrwxrwx 1 root root 5 Mar 31 03:54 file2 -> 
  /home/root/file1
  ```
  
  At the end of the line listing the symbolic link, an arrow indicates the full path of the file pointed by the link.
The \texttt{vi} text editor

- An editor in text mode only
- Very useful to edit files when only a text console is available, for systems with no display facility (embedded systems, typically)
- Command line based management, difficult to learn when used to GUI based editors

Basic commands

- Insertion mode: \texttt{i}
- Erase: \texttt{x}
- Quit + save: \texttt{<esc>} :\texttt{wq}
Compile and run « Hello World » module on your laptop

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("Dual BSD/GPL");

int hello_init(void);
void hello_exit(void);

module_init(hello_init);
module_exit(hello_exit);

int hello_init(void) {
    printk("<1> Hello world!\n");
    return 0;
}

void hello_exit(void) {
    printk("<1> Bye, cruel world\n");
}
```
Training #1

- Compile and run « Hello World » module on your laptop
  - Install kernel headers
    - Know your kernel version
      - `uname -r`
    - Check/install kernel headers if needed
      - `/lib/modules/<kernel_version>/build`
      - `/usr/src/linux-headers-<kernel_version>`
      - `sudo apt-get install linux-headers-<kernel_version>`
  - Compile modules
    - Makefile requires the path to the kernel headers
    - `#!/\ Beware tabulations`
      - `obj-m = hello.o`
      - `all:`
        - `make -C /usr/src/linux-headers-<kernel_version> M=$(PWD) modules`
      - `clean:`
        - `make -C /usr/src/linux-headers-<kernel_version> M=$(PWD) clean`
Compile and run « Hello World » module on your laptop

- Modules execution
  - Install module
    - `sudo insmod hello.ko`
  - List modules
    - `lsmod`
  - Remove modules
    - `sudo rmmod hello.ko`
  - Check kernel messages
    - `dmesg`
    - `dmesg | tail`
Linux Shell

See: [http://linuxcommand.org/tlcl.php](http://linuxcommand.org/tlcl.php) Learning the Shell / Manipulating files

**Exercise 1**
Search in the home directory and return all files ending by `.c` (using **find** command)
Search in the home directory and return all files starting with `X` or `x`
Search in the home directory and return all files starting by a numeric character

**Exercise 2**
Find the number of files in `/usr/bin` which size exceeds 1Mo

**Exercise 3**
Add directory **essai-grep** in your home directory. In this directory, create the following files: **tomate** **poire** **pomme** **cerise** **Fraise** **fraise** **courgette** **POMME3** **afraise**
Using `ls | grep`, list the files of this directory with the following constraints:
Constraint 1: the file name must be Fraise or fraise
Constraint 2: « se » is at the end of the file name
Constraint 3: « ai » is in the file name
Constraint 4: file name contains one numeric character
Constraint 5: file name contains string « mm » or « MM »
Linux Shell

See: [http://linuxcommand.org/tlcl.php](http://linuxcommand.org/tlcl.php) Learning the Shell / Manipulating files

**Exercise 1**

```bash
find /home/bilavarn/ -name "*.c"
find /home/bilavarn/ -name "[Xx]*"
find /home/bilavarn/ -name "[0-9]*"
```

**Exercise 2**

```bash
find /usr/bin/ -size +1M
find /usr/bin/ -size +1M -ls
```

**Exercise 3**

```bash
mkdir essai-grep
cd essai-grep
touch tomate poire pomme cerise Fraise fraise courgette POMME3 afraise
Constraint 1 : ls | grep "^[fF]raise"
Constraint 2 : ls | grep "se$"
Constraint 3 : ls | grep "ai"
Constraint 4 : ls | grep "[0-9]*"
Constraint 5 : ls | grep "^[mM]\{2\}"```