Basics of Mobile Linux Programming

Matthieu Weber (mweber@mit.jyu.fi), 2010

Plan

- Linux-based Systems
- Scratchbox
- Maemo
- Meego
- Python

Linux-based Systems

What Linux is Not

- An operating system: Linux is only a kernel
- A washing powder (in the context of this course)

A Short History

- 1983: GNU project ("GNU is Not Unix")
- 1987: Minix (as a teaching tool)
- 1991: Linux kernel in Minix 1.5
- 1992: Linux becomes GNU's kernel (Hurd is not ready)

Relation to Unix

- Unix was developped in 1969 by AT&T at Bell Labs
- Many incompatible variants of Unix made by various manufacturers
- Standardisation attempt (POSIX) in 1988
- Linux implements POSIX (kernel part)
- GNU implements POSIX (application part)

Distributions (1)

- Linux distributions are made of
  - a Linux kernel,
  - GNU and non-GNU applications,
  - a package management system
  - a system installation tool
- Examples: RedHat EL/Fedora Core, Slackware, Debian, Mandriva, S.u.S.E, Ubuntu

Distributions (2)

- Distributions provide a comprehensive collection of software which are easy to install with the package management tools
- Packages are tested to work with each other (compatible versions of libraries, no file-name collision...)
- Packages often have explicit dependencies on each-other ⇒ when installing a software, the required libraries can be installed at the same time
- The origin of the packages can be authenticated ⇒ less risk to install malware

GNU General Public Licence

- Software licence for GNU, used by Linux
- Free software licence, enforces the freedom to:
  - run the software for any purpose
  - study/modify the program
  - redistribute source code
  - redistribute modified source code
- Free as in "free speech" not as in "free beer"

Unix Concepts

- Multiuser, multitasking, server-oriented
- Hierarchical file system, single tree with links
- "Everything is a file" (devices, sockets, pipes, directories...)
- "One task, one program" (lots of small CLI tools)
- Configuration files are text files

Filesystem Structure (1)

`/` root directory

`/bin` basic system binaries

`/boot` kernel image and boot-related files

`/dev` device files

`/etc` configuration files

`/home` user directories

Filesystem Structure (2)

`/lib` system libraries

`/sbin` administrator programs

`/tmp` temporary files

`/usr` common application programs sub-tree

`/usr/local, /opt` machine-specific, local application programs

`/var` generated, "variable size" files

Users

- Users are identified by a username (internally by number, the UID)
- Users are granted permissions in the system, mostly regarding files
- The system administrator is called *root* and has UID 0
- Users also belong to groups, identified by names (and numbers, the GID)

File Ownership

- Files always belong to one user and to one group (not necessarily one of the user's groups)
- Applies also to directories, devices, named pipes, sockets, symbolic links...

File Permissions

- Files have 3 main permission categories: Read/Write/eXecute, which are set or not
- Each file has 3 ownership categories: user, group and other
- Each permission category applies to each ownership category: user r/w/x, group r/w/x, other r/w/x

Shell

- Basic user interface (command interpreter), command line oriented
- Also a programming language interpreter (shell scripts)
- Many different shells: `sh`, `ksh`, `bash`, `ash`, `dash`, `zsh`; `csh`, `tcsh`; `sash`
- From the shell, the user calls other programs

Useful Commands

`cd` changes directory

`ls` lists files in the current directory

`mv, cp, rm` rename, copy, delete a file

`mkdir, rmdir` create, delete a directory

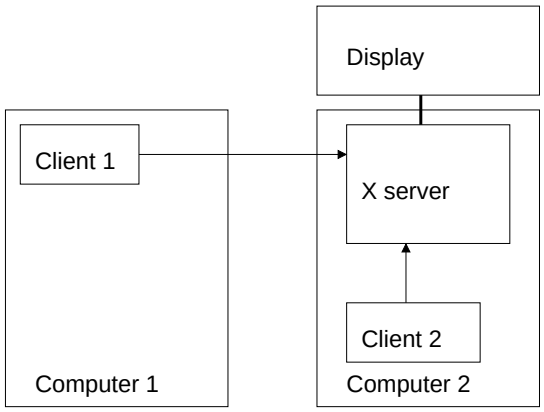
`chmod` changes the permissions

`man` displays a manual page for a command

X-Window (1)

- Graphics-mode display usually provided by X-Window
- The X server is an independant program
- Graphical applications are clients of the X server
- The X server does not need to run on the same computer as the clients
- Handles graphics, fonts and input events (from keyboard, mouse, tablet, joystick...)

X-Window (2)



`DISPLAY` Environment Variable

- Environment variables give information to the current process
- "Exported" variables pass the information to the children of the current process
- The `DISPLAY` variable tells X clients how to connect the X server.
- Examples:
  - `export DISPLAY=:0` tells to connect to display 0 on the local machine,
  - `export DISPLAY=remote.machine.com:1` tells to connect to display 1 on remote.machine.com

Clutter

- Maemo 5 and MeeGo use OpenGL for 3D animations, through the Clutter library
- Clutter allows to display 2D surfaces in a 3D space

Graphical Toolkits

- Libraries providing common GUI widgets (windows, buttons, lists, input boxes...)
- Examples: Athena widgets (Xaw), Motif/Lesstif, Gimp Toolkit (GTK+), Qt, Swing, wxWidgets...
- Maemo 5 is based on GTK+ (legacy) and Qt
- MeeGo is based on Qt (handset variant) and MX clutter (netbook variant, based on Clutter)

Window Managers

- X-Window displays basic rectangular windows with no decoration
- Window managers allow the users to move, resize, hide. . . windows
- Old Maemo used Matchbox, Maemo 5 uses only a library of core features of Matchbox
- Meego uses a window manager of its own

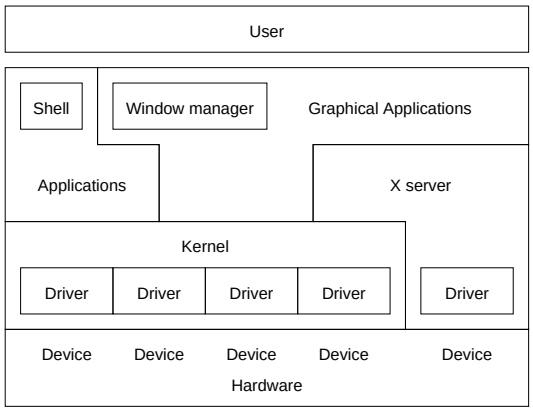
Compositing Window Manager

- Tell X-Window to draw the content of the windows to a hidden buffer instead of the actual on-screen framebuffer
- An external process can then modify this buffer before displaying it on screen
- Allows to add e.g., 3D effects
- Maemo 5 and MeeGo use such a window manager

Remote Computer Access

- Linux/Unix systems can be used interactively remotely:
  - In text mode using e.g., `ssh`
  - In graphics mode using e.g., X-window or VNC

System Diagram



Scratchbox

Presentation

- Cross-compilation toolkit for embedded Linux applications
- Tools to integrate and cross-compile an entire Linux distribution
- Licenced under GPL

Features

- Supports ARM and x86 targets (PowerPC, MIPS, CRIS under development)
- Mainly Debian support
- glibc and uClibc
- QEMU or real target hardware to execute cross-compiled binaries
- Runs with normal users privileges

Structure

- Entire Debian system in one directory per user
- One directory per target
- Common files are shared between targets
- Execution in a chrooted environment

Usage

- Type `/scratchbox/login` in a terminal
- Any command in this terminal runs in Scratchbox with user privileges, including e.g., package installation.
- `sb-conf select DIABLO_ARMEL` selects the ARMEL target; runs ARMEL binaries through QEMU and compiles binaries for the N800 device
- `sb-conf select DIABLO_X86` selects the X86 target; runs x86 binaries straight on the host and compiles binaries for the host

Starting the GUI

- Outside Scratchbox type: `Xephyr :2 -host-cursor -screen 800x480x16 -dpi 96 -ac`
- In Scratchbox type: `export DISPLAY=:2`
- Type: `af-sb-init.sh start` to start the graphical environment
- Type: `run-standalone.sh` *program* to run *program* in the Hildon framework

Maemo

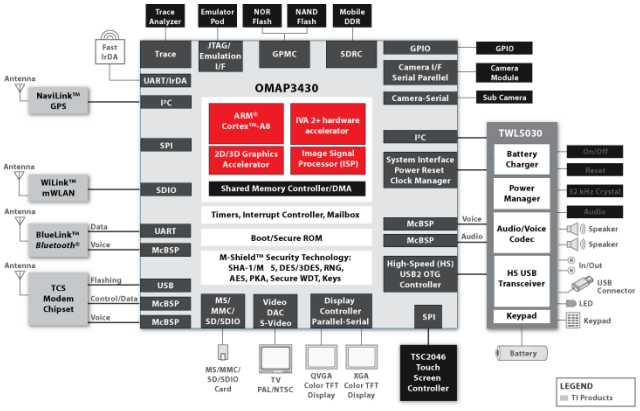
Presentation

- Open source development platform mostly for Nokia Internet Tablets
- Made up mostly of free components, from desktop and embedded systems
- Based loosely on Debian GNU/Linux distribution
- Linux kernel, X-Window (Kdrive), GTK graphical toolkit, Matchbox window manager, Hildon graphical framework (based on Gnome)
- D-Bus, Telepathy, gstreamer, GConf
- Proprietary software for exotic hardware (e.g., camera, WLAN, GPS, software keyboard)

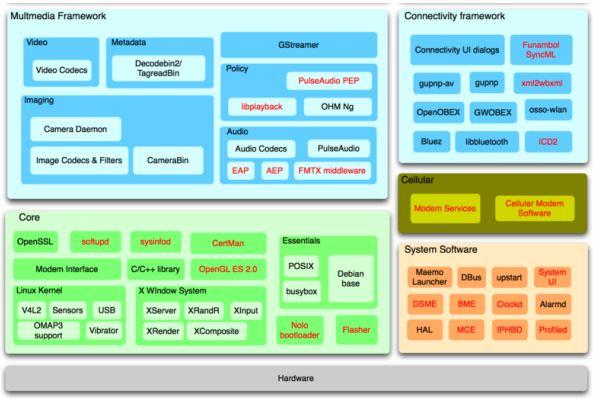
Programing Languages

- Maemo supports C, C++, Python, Perl, bash (and the usual CLI tools: awk, sed... through Busybox)
- Target architecture is ARM11

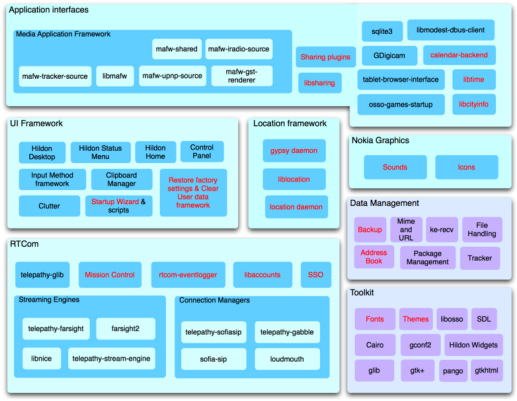
N900's OMAP3430 Processor



Maemo Platform (1)



Maemo Platform (2)



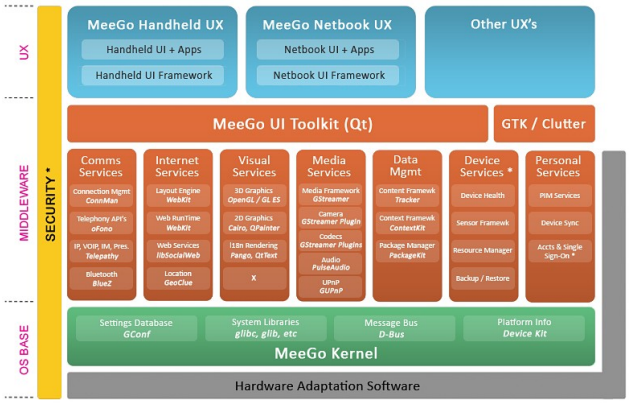
Graphical Interface

Meego

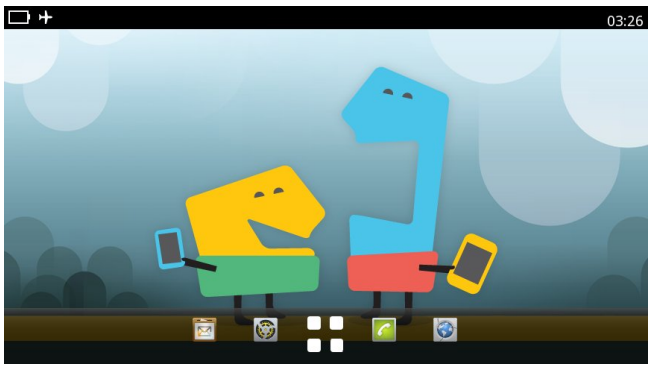
Presentation

- Open source development platform, merge between intel's Moblin and Nokia's Maemo
- Originally for netbooks, nowadays targetting mobiles devices
- Made up mostly of free components, from desktop and embedded systems
- Uses RPM as a package manager, but not based on any particular Linux distribution
- Essentially similar to Maemo 5

Maemo Platform



Graphical Interface



Python

Presentation

- Scripting language
- Developed by Guido Van Rossum since 1991
- Multi-paradigm (functional, object oriented and imperative)
- Fully dynamic type system
- Automatic memory management
- Wide library of free modules

Syntax Features

- Indentation matters, defines blocks of code
- Little use for parenthesis
- Ready-made data types: integer, boolean, float, string, list, dictionnary, tuple

Documentation

- Maemo/Meego use version 2.5
- Latest version is 3.1 (contains radical changes from 2.x)
- Base page: `http://docs.python.org/release/2.5/`
- Tutorial: `http://docs.python.org/release/2.5/tut/tut.html`
- Built-in functions and basic modules: `http://docs.python.org/release/2.5/lib/lib.html`
- Other modules: `http://docs.python.org/release/2.5/modindex.html`

Simple Example

```
def hello(n):
  for i in range(n):
    s = "Hello "
    if i % 2 == 0:
      s += "even (" + str(i) + ")"
    else:
      s += "odd  (%d)" % i
    s += " world !"
    print s

hello(10)
```

Sequences Example

```
string = "abcdefg"
weekdays = ["Mon", "Tue", "Wed", "Thu",
            "Fri", "Sat", "Sun"]
days_of_week = {"Mon": 1, "Tue": 2, "Wed": 3,
                "Thu": 4, "Fri": 5, "Sat": 6,
                "Sun": 7}

print string[2], string [-2]
print weekdays[4], weekdays[-1]
print days_of_week["Wed"]
```

Classes and Methods

- class keyword
- All members are public, except if their name starts with __.
- A class instance is returned when calling the class name
- Methods are objects

Class Example (1)

```
class Divider:
  def __init__(self, divider=1):
    self.divider = divider

  def divide(self, n):
    try:
      self.result = n / self.divider
    except:
      self.result = None
```

Class Example (2)

```
  def __str__(self):
    if self.result == None:
      return "Not a number"
    else:
      return "%d" % self.result

d = Divider(0)
d.divide(4)
print d
```

Modules

- Lots of code provided as extra modules e.g., GTK+
- Modules usage:
  - `import` *module* (e.g., `import gtk`)
    → names must be qualified with the module's name (e.g., `gtk.Button()`)
  - `from` *module* `import` *name, ...* (e.g., `from gtk import Button, Label`)
    → *name* is added to the current namespace (e.g., one can call simply `Button()`)

Exceptions

- Python handles exceptions
- `try` and `except` keywords (and `else`)
- Exceptions are classes too

Exception Handling Example

```
import sys
try:
  f = open('myfile.txt')
  s = f.readline()
  i = int(s.strip())
except IOError, (errno, errstr):
  print "I/O error(%s): %s" % (errno, errstr)
except ValueError:
  print "Converting to an integer failed."
```

Special Methods

`__init__` class instance initialization (constructor)

`__del__` class instance destructor

`__str__` conversion to string

`__lt__, __eq__...` comparison methods

`__hash__` generates a key when the instance is used in dictionary

`__call__` used when the object is called as a function operations

Dynamic Attributes and Introspection

- New attributes can be added at runtime to Classes and class instances
- Attributes can be:
  - deleted (`del()` function)
  - listed (`dir()` function)
  - queried (`hasattr()` function)

Conclusion

- Linux is not only for servers, it's widely used in embedded systems
- Maemo and Meego reuse free software and adapt it to mobile devices
- Python allows quick development of applications on any platform