Introduction to Linux EECS 211

Martin Luessi

April 14, 2010

Martin Luessi ()

Introduction to Linux

Image: Image:

April 14, 2010 1 / 14

#### 1 Introduction

- 2 How to Get Started
- **3** Software Development under Linux
- 4 Hands-on Demonstration

#### What is Linux?

- UNIX like OS which uses the Linux kernel
  - Originally developed by Linus Torvalds in 1991
  - Over 10 million lines of code

The whole OS consists of kernel and user space (originally from GNU project)

Kernel hardware management (device drivers), process and memory management, file systems, networking, ... User space init system, desktop environment, http server, web browser, video player, ...

- Very modular with numerous choices for components which provide the same functionality
- Linux distributions integrate components into a fully functioning OS. Examples for distributions are: Debian, Red Hat, SuSE, Gentoo, Ubuntu, Arch, ...

# Why Should You Care?



- The Linux ecosystem consists of free software
  - You are free to use it for any purpose
  - You are free to study how it works, and modify it to make it do what you wish
  - You are free to redistribute copies and modified versions
- It is important to be familiar with software development for Linux systems, as they are very widespread:
  - 60% of all web servers run Linux
  - 446 out of the top 500 fastest supercomputers run Linux
  - Intel and Nokia are working on "MeeGo", which will run on a wide range of consumer devices (cell phones, netbooks, in-vehicle...)
  - Google's Chrome OS is Linux based
  - Android uses the Linux kernel
- Looks good on your resume, many job opportunities





How to Get Started

#### How to Get Started - Overview

- Choosing a Linux distribution
- Installation method
- First steps

#### How to Get Started

## Choosing a Linux Distribution





There are many distributions (google for "Linux family tree")

- Do some research, criteria include:
  - Ease of installation and use: Mainstream distributions (Ubuntu, Fedora, SuSE, Mandriva,...) are easy to use but allow for little customization (and you will learn less)
  - Is the distribution actively developed?
  - How many software packages are available?
  - Which desktop environment(s) does it support?

Approximate Process by Early 10-40	📈 🧰 🛃 🤘 DA DA 13, BAD	
	0 WHEN PERSON A PARTY	00
Contract Line	And a rest of postate be large	e v
The Dis line line (b)	E in the regent of doors will see the set of the second se	
• Name v	2 Mail 1990 11 Mail	et features same
portection g     proceeding the angle	GHOME Decumentation Library	toolgon dest
· many research	GNOME 2.28 Release Notes	UNDER 2 28 Trabase
Encircle Method aggle autors     Encircle Method aggle autors	Interdiction	Andreas Service     A
	L. Introduction 0004.1.19.5 the next version of the MEMI Institute a produce multipletion advances of the year response. ODDR's have a house of how, statisty, and then the intervention of the control of a control of the year. ODDR's the state of the state of the state of the year of the year of the state of the state of the year of the year.	
1111	File and riper hards Minary and provide and if the commonlexit compare scale report of a manifest comparing on minary scale and the scale programmers with transmiss the minary scale in the scale programmers with transmiss the provide a field and scale and in address to a scale scale devicement with the scale and scale and in address to device and the scale and scale and in address to the scale and the scale and the scale and the scale and the devicement of the scale and the scale and the scale and the devicement of the scale and the scale and the scale and the devicement of the scale and the scale and the scale and the devicement of the scale and the scale and the scale and the devicement of the scale and the scale and the devicement of the scale and the devicement of the scale and the devicement of the	100 000.000



#### Installation Method

There are several ways you can "install" Linux:

- Use Linux inside a virtual machine
- Boot from CD/DVD or USB stick (for some distributions, e.g., Knoppix)
- Dual boot (requires changing the partition layout)
- Use Linux as primary OS (Windows inside a virtual machine)

#### First Steps

- Become familiar with the desktop environment
- Familiarize yourself with the directory structure
- Learn how to install additional software using a package manager
- Learn how to use the shell (terminal)
  - In many cases more efficient than using a GUI program
  - Automate tedious things using shell scripts
  - TAB completion is your friend



# **GNU** Toolchain

- The GNU toolchain is most commonly used for C/C++ development under Linux
- The important parts for us are:

GCC Suite of compilers for several programming languages (GNU Compiler Collection)

GNU make Automation tool for compilation and build GDB Debugger (GNU Debugger)

# GCC - GNU Compiler Collection

- Supports a large number of programming languages (C, C++, Java, Ada, Objective-C, Objective-C++, Fortran..) and platforms (x86, x86-64, ARM, MIPS, PowerPC..)
- Compiling a C++ program consisting of a single source file is very simple:

```
g++ main.cpp -g -o main
```

This compiles the source code directly into an executable

- Larger projects consist of multiple .cpp files
- To build an executable, each .cpp is first compiled into an object file

   (.o) which are then linked together to obtain an executable
- GNU make helps automating the build process

## GNU make & Makefiles

Make uses makefile(s) which specify how to obtain a target program from each of its dependencies

The basic syntax for an entry is

```
<target>: [ <dependency > ]*
[ <TAB> <command> <endl> ]+}
```

For the previous example a simple makefile could be

main: main.cpp <TAB> g++ main.cpp -g -o main

For one source file using a makefile does not make too much senseMake is mostly useful when using multiple .cpp files

#### GNU make & Makefiles Cont.

Assume you have 2 source files main.cpp and file\_reader.cpp, in this case the makefile could be

```
main: main.o file_reader.o
<TAB> g++ main.o file_reader.o -o main
main.o: main.cpp
<TAB> g++ -c main.cpp
file_reader.o: file_reader.cpp
<TAB> g++ -c file_reader.cpp
```

- A makefile like this quickly becomes tedious to maintain for larger projects
- Solution: use macros

#### GNU make & Makefiles Cont.

A more sophisticated makefile using macros:

```
CXXFLAGS = -02 -g -Wall
OBJS = main.o file_reader.o
LIBS =
TARGET = main
$(TARGET): $(OBJS)
<TAB> $(CXX) $(CXXFLAGS) -o $(TARGET) $(OBJS) $(LIBS)
all: $(TARGET)
clean:
<TAB> rm -f $(OBJS) $(TARGET)
```

The (builtin) CXX macro assumes that there is a .cpp file for each object file, it automatically compiles each .cpp file and links the object files together

# Hands-on Demonstration

- Using Ubuntu in a virtual machine (VirtualBox)
- Desktop environment
- Installing software (CodeBlocks, GCC, GUN make, etc)
- Using CodeBlocks
- Sharing files with the host OS
- Using the shell and other useful tools