

CSE 303
Concepts and Tools for
Software Development

Magdalena Balazinska (a.k.a. “Magda”)

Winter 2010

Lecture 1 – Course Introduction

The Goal of 303

- Learn to write cryptic stuff like this (1 week)

```
if [$# -lt 3] then ... fi
f1=$1; f2=$2
if [[ -a "$f1" && -a "$f2" ]] then ... fi
```

- Or like that (3 weeks)

```
char ** ans = (char**)malloc(height*sizeof(char*));
int i;
for(i=0; i < height; ++i)
    ans[i] = (char*)malloc(width*sizeof(char));
```

- And say things such as

“I **checked out** the project but could not **commit** my changes because the subdirectory in **cv**s was not **group writable**. ” (a few weeks)

More Seriously...

Main Goals of CSE 303

- Put you on the path to becoming a **mature and efficient software developer**
- Make your life easier in subsequent classes, internships, research projects, and jobs
- Raise your sensibility to the societal and ethical implications of software systems
- In the past, software developers had to acquire *on their own* the skills and concepts taught in cse303

Outline for Today

- Class mechanics
 - Staff and resources
 - Lectures
 - Assignments and evaluation
- Overview of topics and class schedule
- General advice
- Introduction to Linux, the filesystem, & shell

Class Resources

- Your number 1 resource: course website

`http://www.cs.washington.edu/303/`

- **Mailing list:** announcements and other info
 - **You should already be registered to the list**
- **Staff:** office hours posted on course website
 - Michael Ratanapintha (TA)
 - Magdalena Balazinska (instructor)
- **Computing resources:** undergrad lab
- **Message board:** link on course website

Lectures

- Three lectures per week: MWF @ 10:30-11:20
 - Introduce important **concepts and tools**
 - Point to **additional readings**
 - We do not expect you to take exhaustive notes
 - Participate & jot down keywords to look-up later
 - Class material posted online after lecture
 - **Advice:** use class for concepts and documentation/books for details
- Remember: This is a 300-level course!

Assignments

- Due soon after content is covered in class
- Spend most of your time on assignments
- Work on each assignment in several sessions
 - Because you will be using new tools...
 - You will feel a constant energy drain...
 - You will sometimes get stuck and need to look up documentation or go to office hours
 - Again, remember that this is a 300-level course!
- Note: We expect almost everyone to get 100%

Evaluation

- 25% Midterm: February 12th in class
- 25% Final: Monday, March 15th in class
- 40% Assignments: total 6
 - 2 on linux, shell scripts, and utilities
 - 2 on C and tools
 - 1 on C++
 - 1 on software engineering and tools
- 10% Issue paper on society and ethics

More About the Assignments

- All assignments can be done in *groups of up to two*
- Collaboration policy between groups
 - Books, lecture notes, manpages, the web
 - You can point each other to *documentation*
 - BUT **each team must produce their own solution**
 - **You may NOT look at solutions of other groups**
- **Late policy:** total of three late-days that you may use anytime in chunks of 24 hours
- **Extra credit:** small effect on your grade

Overview of Assignment 1

- HW1 will be posted on website this Wednesday
- Due date: Wednesday, January 13th
- Assignment content
 - Try various programs and options
 - Try a few useful shortcuts
 - Try using man and Google
 - Write a simple shell script
- Use office hours this week or next week for help logging in and getting started!

Where to Go for Information

- Required texts:
 - **Linux Pocket Guide** by Daniel J. Barrett, O'Reilly, 2004.
 - **Programming in C (3rd Edition)** by Stephen G. Kochan, Sams Publishing, 2005.
- Course website
 - Lecture notes
 - Links to additional documentation
- A lot of information is available on [the web](#)
- [Manpages](#)

That's it for the class logistics...
now let's take a look at the class content

Five High-Level Topics

- Expedite and automate tasks
 - Become familiar with **Linux** and various **utilities**
 - Manipulate files and strings
 - Write shell scripts: **bash**
- Learn to program in C
 - “Lower level” than Java
 - Emphasis on memory management and pointers
 - A little bit of C++ to get you started
 - A taste of threads and concurrency control

Five High-Level Topics

- Learn basic tools for software development
 - Build tools (compiling, linking, and automating)
 - Debuggers
 - Version control systems
 - Profilers (if we have time)
- Acquire basic software engineering concepts
 - Specifications, interfaces, and testing
 - Multiperson programming
 - Security and defensive programming

Five High-Level Topics

- Societal and ethical implications of software
 - Because technology affects society
 - As professionals/scientists/engineers, we must understand societal implications of what we do
 - 4 in-class discussions
 - Topic will be announced before the class
 - Examples: software patents, digital privacy, digital rights management, electronic voting, etc.

Course Schedule

- Posted schedule subject to small changes
- Visit course website regularly

General Advice

- We will continuously learn new tools
 - We will barely scratch the surface for each tool
 - The goal is to get you started and help you learn
 - You may constantly feel a certain unease
- Lectures alone are not enough
 - Books and documentation provide details
 - Assignments give you practice
- Work on class a little bit after each lecture
 - Assignments due soon after we cover material!
 - Enjoy it when you get something to work!

The Good News

- We assume you don't know much, just some Java programming and some simple data structures
- So ask questions
- Now is the best time!

Summary

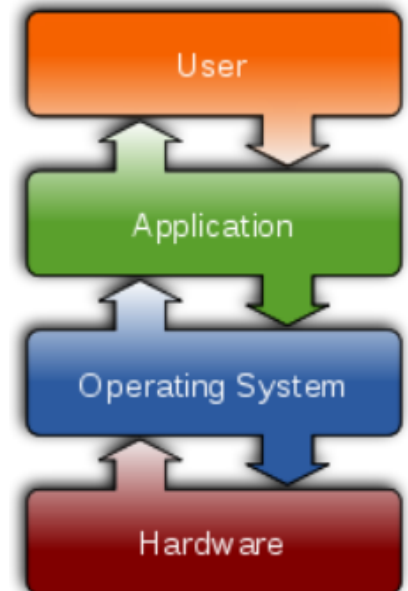
- Goal: maturity and efficiency
 - Command-line
 - C/C++
 - Programming tools
 - Software-development concepts
 - Social/ethical implications of computing
- This class is just the beginning
- You will learn throughout your career

That's it for the class introduction.
We have a lot to cover this quarter...
so let's get started

Operating System

- **Operating system:** Manages activities and resources of a computer:
 - Software that acts as an interface between hardware and user
 - Provides a layer of abstraction for application developers

- **Features provided by an operating system:**
 - Ability to execute programs (and multi-tasking)
 - Memory management (and virtual memory)
 - File systems, disk and network access
 - An interface to communicate with hardware
 - A user interface (often graphical)



http://en.wikipedia.org/wiki/Operating_system

- **Kernel:** The lowest-level core of an operating system

Linux

- **Linux:** An operating system based on the linux kernel
 - Unix-like
 - Commonly seen/used today in servers, mobile/embedded devices, ...
- A full Linux operating system usually comprises
 - Utilities and libraries from the GNU Project
 - The X Window System
 - The GNOME and KDE desktop environments
 - The Apache HTTP Server
- **Distribution:** A pre-packaged set of Linux software
 - Examples: Ubuntu, Fedora
- Key features of Linux
 - **Open source software:** source can be downloaded
 - Free to use
 - Constantly being improved/updated by the community



Let's Start at the Beginning

- First, log in with user name and password
- You will get a **shell**
- What is a shell?
 - *Program that works with the OS as a **command** processor, used to enter commands and initiate their execution.*
- Typically, a **command** is a program name with options and argument: `ls -al dirA`
- The shell also provides “built-in” commands:
`cd ..`

Exploring the Filesystem

- The filesystem is a tree (rather a dag)
 - The top is `/`
 - Interior nodes are directories
 - Directories contain files and subdirectories
 - Moving around: `cd`
 - Got lost? `ls` and `pwd`
- Each user has a home
 - Typically it is in: `/home/username/`
 - But it can be somewhere else

Continuing to Explore...

- Special directory names

- Root directory = /
- Current (working) directory = .
- Parent directory = ..
- User's home directory = ~

- Paths

- Absolute pathname starts from the root

`/home/username/dirA`

- Relative pathname starts from current directory

`~/dirA` or `../dirA`

Permissions

- Permissions (read, write, execute)
 - Your user name determines your permissions
 - Different permissions for a user and for everyone
 - Users sometimes work together in a **group**
 - **Changing permissions:** `chmod`

Basic File Manipulation

- Examining files

`cat, head, tail, less`

- Creating and destroying

- Files: `cp, mv, rm, rm -f`

- Directories: `mkdir, rmdir, mv, cp -r`

Commands and Programs

- It helps to remember important commands
 - `ls`, `cd`, `pwd`, `cp`, `mv`, `rm`, `mkdir`, ...
- Many commands correspond to programs
 - `ls`, `pwd`, `cp`, `mv`, `rm`, `mkdir`
- Some commands are shell “builtins”
 - `cd`, `echo`, `exit`
- Use `type` to distinguish them
- A running program is a process
 - (could be more than 1)

Why Use a Shell?

- I can do all this with a GUI. Why use a shell?
- Power users can go **faster** with a shell
- Simpler and faster when logging in **remotely**
- Enables task automation: **programmability**
- Enables **customization** of linux session
- Most computer scientists use both
- Windows and Linux provide both

Shell Scripts

- Series of individual commands combined into one executable file form a shell script
- Shell is an interpreter for a programming language of the same name
 - Variables
 - Some prog. constructs: conditional, loops, ...
 - Integer arithmetic
 - etc.

Readings

- **Sections from the Linux Pocket Guide**
 - What's in This Book (pages 1-5)
 - Getting Help (pages 7-8)
 - In the Filesystem section
 - Introduction (page 13)
 - Home Directories (pages 14-15)
 - File Protections (pages 19-20)
 - The Shell (pages 19-33)
 - Skip subsection on Installing Software
 - Pages 37-46 give more details about the commands that we used today