COURSE INTRODUCTION

Software Tools and Systems Programming
CSC209 Winter 2015
Bogdan Simion

Thanks to Karen Reid and Alan J Rosenthal for (most of) the material in these slides
What CSC209 is about?

- The environment in which your programs run
  - understanding the environment
  - developing tools:
    - for interacting with the environment
    - for getting information about it
    - for influencing it
  - learning a new language (or two) to help us
Course Overview

- **Part I (~2 Weeks)**
  - UNIX Environment
  - Understanding the Shell
  - Shell Programming (deferred to later in the course)

- **Part II (~3-4 Weeks)**
  - C Programming: Basics, Arrays, Strings, Files, Structures, Dynamic Memory Management

- **Part III (~6 Weeks)**
  - Processes, Pipes, Signals
  - Sockets and Network Programming
  - Concurrency
UNIX gives you building blocks

- In this course you will learn how to use them

Self Study Topics

- Using Unix - some tutorial coverage
- Using software tools
  - an editor – vi/vim, emacs, nedit, …
  - an IDE – eclipse, …
  - a debugger – gdb, …
- Readings
Environment

- Environment: Computing Disciplines Facility (CDF)
  - LINUX system
  - cdf.utoronto.ca / cdf.toronto.edu
    - ssh username@cdf.toronto.edu
  - Use your CDF login and password
  - NX server (see CDF website)
  - Using CDF: http://www.cdf.toronto.edu
  - CDF labs: BA3175, BA3185, BA3195
Windows & Mac Users

Windows: If you want to do some of your work on your own machine, two options:

- Install cygwin: www.cygwin.com
  - Only emulates Unix tools, possible problems, don’t recommend it!
- Use a virtual machine:
  - VirtualBox: www.virtualbox.org
  - VMWare (commercial, VMPlayer free version): www.vmware.com/ca/en

MacOS: UNIX flavour, use “Terminal” application

Always double-check assignments on CDF too!
Become familiar with Linux – it’s neat!

Source: http://www.bambiltech.com/2009/01/26/12-wallpapers-in-which-linux-criticizes-windows/
For my interest

- How many of you have LINUX knowledge?
- How many of you have done some shell scripting?
- How many of you have programmed in C or had attended an introductory course in C?
Today’s Overview

- Course Administrivia
- Unix Overview
- The Big Picture
CSC209 Administrivia
Course Information

- **Instructor:** Bogdan Simion
- **Lectures (WI1016):** Mon-Wed 10:00-11:00
- **Office Hours (BA2230, for now):** Mon-Wed 11:00-12:00
- **Tutorials (CDF labs):** Fri 10:00-11:00
- **Email:** bogdan@cs.toronto.edu
- **Course Website:** [http://cdf.toronto.edu/~csc209h/winter](http://cdf.toronto.edu/~csc209h/winter)
Communication

- Discussion board: Piazza (see website)
  - General questions about course material, clarifications about the content
  - Questions about assignments, tutorials, midterm, final, etc.
  - Anything of general interest for everyone in the class

- Email:
  - For individual problems (prerequisite waiver requests, remarking requests, doctor’s notes, etc.)
  - Subject must include **CSC209** (or **csc209**)
  - Email is a formal method of communication:
    - State your question clearly, with enough context
    - **Sign it** (Name, login and student # are the most useful)
Course Textbooks

Course Prerequisites

- Make sure you have the prerequisites!
  - CSC207 - Software Design

- If you don’t have the CSC207 prerequisite, ask me for a waiver by email (no guarantees though!)
## Course Marking Scheme

<table>
<thead>
<tr>
<th>Work</th>
<th>Weight</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>4%</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>8%</td>
<td>C basics</td>
</tr>
<tr>
<td>A2</td>
<td>10%</td>
<td>C advanced</td>
</tr>
<tr>
<td>A3</td>
<td>10%</td>
<td>Processes, pipes</td>
</tr>
<tr>
<td>A4</td>
<td>8%</td>
<td>Socket programming</td>
</tr>
<tr>
<td>1 Midterm Test</td>
<td>15%</td>
<td>March 4th</td>
</tr>
<tr>
<td>Final Exam</td>
<td>45%</td>
<td><strong>You must get &gt;=40% to pass the course</strong></td>
</tr>
</tbody>
</table>
Assignment Policies

- Assignments are due at 10:00 p.m. on the due date - check website for final due dates
- Late Assignment Policy: 3 grace days
- Code must work on CDF servers
- Make sure you commit all your source files; we cannot find files you never submitted
- Code style matters!
- Test-as-you-go
- The code you submit has to work, even if it doesn’t implement everything
- Code that does not compile gets zero marks
Did you catch that?
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
I will not submit code that does not compile!
Submitting Assignments

- You will be using the **MarkUs** system to submit your assignments
- Details will be provided on how to submit your assignments
- **Start early on the assignments!**
- Make sure you can commit in your **svn repository** and commit periodically
- **Do not wait until the very last minute to submit your assignment!**
Plagiarism

- “The work you submit must be your own, done without participation by others. It is an academic offense to hand in anything written by someone else without acknowledgement.”
- You are not helping your friend when you give him or her a copy of your assignment.
- You are hurting your friend when you ask him or her to give you a copy of their assignment.
What is Cheating?

- Cheating is
  - copying parts or all of another student’s assignment
  - including code from books, web sites, other courses without attribution
  - getting someone else to do substantial parts of your assignment
  - giving someone else your solution

- Cheating is not
  - helping to find a bug in a friend’s code (be careful)
  - helping each other understand man pages or example code
A few do’s and don’ts

- **Do**
  - ask questions if you don’t understand something
  - work together to understand concepts/assignments
  - use tutorials and office hours
  - read textbook or provided online material before class

- **Don’t**
  - hand in other peoples’ work (it’s cheating)
  - harass others (see the University’s policies)
  - distract or disrupt the class (it’s immature)
UNIX is user-friendly. It’s just very selective about who its friends are.
Brief Unix History

- Developed in 1969 by a group of AT&T employees at Bell Labs (Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk and Joe Ossanna)
- Pipes and C (successor of B) were added in 1971-73
- “License to universities, but no support.”
  - This led to extensive sharing and collaborations
- University of Toronto on the first mailing list in 1975
  - Canadians: Brian Kernighan, Rob Pike, ...
- Berkeley Software Distribution (BSD) => FreeBSD, NetBSD
- 1991, Linus Torvalds (Linux kernel initiator) describes his experimental OS modeled on Minix (Unix-like OS)
UNIX-based systems nowadays
Why Unix?

- Multi-user, multi-tasking OS
- Available on many platforms
- Shares computer resources sensibly
- Allows manipulation of files, processes, and programs
- Allows inter-process and inter-machine communication
- Allows access to its operating features
Ways of Looking at a System

- Some of the ways we look at UNIX:
  - As an end user
  - As an environment for programs to run
  - As a file system - part of the overall environment
UNIX: End-user Interaction

- Unix has a rich set of tools for dealing with its own structures and data:
  - need to be familiar with them to manage (your portion of) the system
  - you may already know some (i.e., move around filesystem, list, copy and remove files, run programs and performing other tasks)
- Involves learning how to write UNIX shell scripts
UNIX: Environment for Programs

- How programs get ready to run
- What happens when a program is run
- What happens when your program writes to/reads from a file
- How your code can start other pieces of code and interact with them
  - how programs "talk" to each other
  - how programs "talk" to the outside world (networks)
UNIX: As a File System

- What are files? what are directories?
  - How are they organized, maintained?
  - What information is accessible about them?

- Everything is a file!
  - Regular files, directories, links, pipes, sockets
  - Devices
    - video (block)
    - keyboard (character)
    - sound (audio)
    - network (block)

- How to access them?
  - File interface: open, read, write, close
THE BIG PICTURE
The Big Picture (in Java)

Source code file
Hw.java

```java
class Hw {
    public static void main(String [] args) {
        System.out.println("Hello");
    }
}
```

Compile it
% javac Hw.java

Object file
Hw.class

Run it in VM
% java Hw

Process
VM
  Hw
The Big Picture

Source code file

```c
#include <stdio.h>
void main()
{
    printf("Hello world");
}
```

Compile it

```
gcc -o hw hw.c
```

Object file

```
hw
```

Run it

```
hw
```
The Shell

% gcc –o hw hw.c

- The % is a shell prompt
- The shell is a program that can execute another program
- The shell
  - accepts commands (programs) as input
  - finds the executable
  - interprets the arguments
  - starts executing the command
- The shell also has some “built-in” commands
Source Code Files

hw.c
#include <stdio.h>
void main()
{
    printf("Hello world");
}

- What is a file?
  - Sequence of bytes

- A file system?
  - A hierarchy of files + tools

- How does the system know where to find hw.c?
  - paths, working directories, ...

- What is the meaning of #include<stdio.h>?

- What does printf really do?
A compiler is a program that translates source code into object (machine) code.

Here we are running the compiler at the command line.

New process!

% gcc -o hw hw.c
Running a program

% gcc -o hw hw.c

After we have compiled the program, we can run it

% hw

load a program into memory and hand it off to the OS that takes control of running it
A Different Big Picture

- Unix system services (system calls)
- Unix kernel (in C)
- computer

libc – C Interface to Unix system services

- sh
- less
- vi
- perl
- gcc
- nedit
- grep
- ddd
Processes

- A **process** is an executing instance of a program
- Multiple instances of a program (e.g., gcc)
- The OS keeps track of information about the process
  - process ID – a unique non-negative integer
  - process state – “running”, “ready”, “blocked”
  - program counter – which instruction is being executed
  - a list of open files
  - etc.

**WHY?**
Program layout in memory

- Typical memory layout of programs.
- The kernel keeps a PCB for each process.
What is Next?

- Shell & Shell Tools

- Remember: Familiar with Linux, you must get!