Course and Unix Intro

Comp-206 : Introduction to Software Systems
Lecture 1

Alexandre Denault
Computer Science
McGill University
Fall 2006

Instructor – Alexandre Denault

- Graduate student, working in the Software Engineering lab.
- Specializes in teaching framework and distributed environments.

- Email: alexandre.denault@mail.mcgill.ca
- Office: McConnell 322 (cubicle in the back)
- Office Hours:
  - Tuesday & Thursday 1h00 - 2h30
  - or send me an email

Official Course Description

Comprehensive overview of programming in C, use of system calls and libraries, debugging and testing of code; use of developmental tools like make, version control systems.

My Course Description

- Introduction to Software Systems ...
  - ... is a course about the various types of tools we use to build software.
  - ... serves as your introduction to the C programming language.
  - ... gives you tools to be more productive during your undergraduate studies.

Course Content

The course will cover the following topics:

- Unix operating system
- Shell Scripting
- C Programming (and related tools)
- Debuggers and Profilers
- Source Control
- HTML, CSS and CGI
- Perl and Web scripting
- Python and GUI

Tentative Schedule

- 4 lectures on Unix and Shell scripting
- 10 lectures on C programming and related tools
- 2 lectures on HTML, CSS and web servers
- 3 lectures Perl programming
- 3 lectures Python programming

Lecture Schedule and Prerequisites

- Lectures:
  - Tuesday and Thursday, 4h05-5h25
  - Macdonald Engineering Building 279
- Prerequisites:
  - COMP 202 or
  - COMP 250

Knowledge of Unix

- This course assumes you have some basic knowledge of Unix (and the school labs).
  - login into your account, copying a file, editing a file, etc
- If you don't, I highly suggest you attend one of the SOCS Unix seminars, to be held in the Trottier Building, 3rd floor, Room 3120 (Lab 2).
  - Beginner: 10:00 - 11:00, Monday, Sept. 11
  - Beginner: 10:30 - 11:30, Tuesday, Sept. 12
  - Beginner: 14:30 - 15:30, Wednesday, Sept. 13
  - Beginner: 14:30 - 15:30, Thursday, Sept. 14
  - Intermediate: 10:30 - 11:30, Monday, Sept 18
  - Intermediate: 10:00 - 11:00, Tuesday, Sept 19

Workload and Grade Distribution

- This course features assignments which require a lot of programming.
- This allows you to put in practice the material learned in class.

- Grade Distribution:
  - Homework Assignments (4) : 40%
  - Midterm : 20%
  - Final exam : 40%

Assignments

- Allows you to practice the material seen in class.
- Allows me to evaluate what you have learned.
- Each assignment is worth 10% of your grade.

- Tentative dates:
  - Assignment 1 : September 18th - October 3rd
  - Assignment 2 : October 3rd - October 24th
  - Assignment 3 : October 24th - November 14th
  - Assignment 4 : November 14th - December 5th

- You lose 15% per late day.
- The T.A. will correct the assignments.

Midterm

- The midterm will allow me to see if you understand the material, before testing you in the final.
- If we didn't see it in class, it's not in the midterm.

- Tentative date:
  - Thursday October 19th, 2006

Academic Integrity

McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offenses under the Code of Student Conduct and Disciplinary Procedures (see http://www.mcgill.ca/integrity/ for more information).

Textbook

- Required Textbook:
  - The C Programming Language (2nd Ed), by Kernighan & Ritchie, Prentice-Hall, ISBN 0131103628
  - Just Enough Unix (5th Ed), by P.K. Anderson, McGraw Hill, ISBN 0072952970
- Recommended textbooks:
  - GNU Software, by Mike Louksides & Andy Oram, O'Reilly Media, ISBN 1565921127
  - A Little Book on Perl, by Robert Sebesta, Prentice-Hall, ISBN 0139279555
  - Dive into Python, by Mark Pilgrim, Apress, ISBN 1590593561

What is Software?

- Software is a collection of instructions (often grouped as functions and libraries) that allow a computer to complete a specific task.
- Although these instructions can be written in different languages, they are eventually converted to a machine language.

Computer Programming

- Computer Programming is the craft (art) of writing the computer instructions that accomplishes a specific task.
- The difficult of computer programming depends heavily on the task itself and the tool that is used.
- One of the key to being a successful programmer is knowing which tool to use in different situation.
  - Not all programming language were designed to do the same thing.

Name that Programming Language

- Java
- C
- C++
- Basic
- Ada
- Lisp
- Scheme
- Javascript
- AppleScript
- Ocaml
- D
- and so on ...

What is an Operating System?

- An operating system is a piece of software that allows us to interact with a computer without having to know the inner working of a computer.
- Its primary function is to manage the computer's resources.
- An operating system also provides us with libraries to interact with these resources.

Name that Operating System

- Dos
- Windows
- Solaris
- Linux
- FreeBSD
- BeOS
- FreeDos
- HP-UX
- AIX
- MacOs X

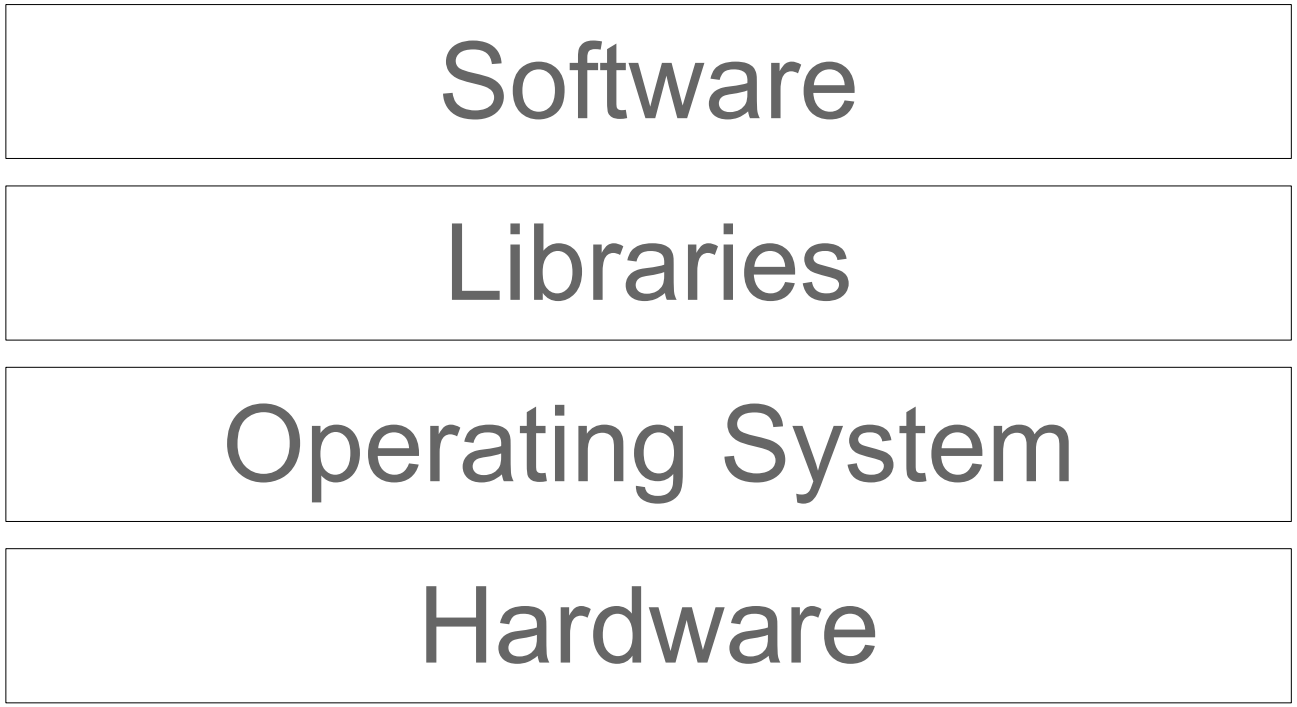
What is a Library?

- A library is a piece of software specially packaged to be used by *other* software.
- A library provides specific functionalities, thus avoiding the need for the programmer to build the functionalities himself.

Library Examples

- On Unix, the Standard IO library provides functionality to open files and write to them.
  - Without it, a programmer would need to write code to use the hard disk itself.
- On Windows, DirectX allows a developer to write display graphics.
  - Without it, a programmer would need to write code to access the video card directly.

Libraries and the OS



Services provided by the OS

- Process management
  - Allows applications to run simultaneously
- Memory management
  - Manages the allocation of memory
- Disk and file systems
  - Manages the writing to disk of files.
- Networking
  - Allows inter-computer communication.
- Security
  - Provides authentication, privacy and protection.
- Device Drivers
  - Allows the use of hardware in a generic fashion.
- Graphic User Interface*
  - Provides a visual interface to interact with the computer.

Process management

- The first generation of operating systems only allowed one process to run at a time.
- A multi-tasking operating allow multiple task to run at a time.
  - Single most computers only have 1 CPU, the multiple task must share this CPU.
  - Most often, the CPU will time-slice the task, so they get a fair amount of CPU time.
- Computers with multiple CPU can run multiple task at a time.
  - This introduces numerous concurrency and coherence challenges.

Memory Management

- At the hardware level, computer memory is a collection of 0 and 1, stored in volatile silicon chips.
- At the operating system, computer memory is an array of bytes, where data can be store and latter retrieved.
- The operating system is responsible for allocating blocks of memory to the different processes.
- When the OS runs out of memory, it can use the hard disk to temporarily store blocks of memory that it does not often use.

Disk and file systems

- At the lowest level, a computer disk is a collection of 0 or 1 stored on a magnetic or optical disk.
- This hardware has no understand of files and directory.
- The operating system provides file systems, which describe how files and directory should be stored on the disk.
- Each operating feature different file system:
  - Windows : Fat32, NTFS
  - Linux: EXT2, EXT3, XFS, Reiser, etc
  - MacOS X: HFS+

Networking

- At the hardware level, networking is the exchange of 0 and 1 over a communication channel.
- A network protocol describe how data should be transformed (and encapsulated) before being sent over the network.
- The OS provides the necessary tool for a computer to join a network and transmit(receive) data over this network.
- The most common network protocol is TCP/IP.

Security

- Operating Systems feature authentication system which control who can use a computer.
- They also offer privacy features on many levels:
  - A user/administrator can define which files can be used by which users.
  - A process should not be able to access the memory block of another process.
- A good operating system will also monitor itself to prevent action which could compromise it's stability:
  - A process should not be able to delete a file which is currently in use.
  - Certain spaces in memory should never be overwritten

Device Drivers

- Each devices connected to the computer have their own different way of communicating with the computer.
- Device drivers are small pieces of software which allow an external device to communicate with the computer in a generic fashion.
- Without device drivers, an application would need to deal separately with all the different external hardware it supports.

Graphic User Interface

- A graphic user interface (also known as a GUI) is a piece of software which allow interaction with the computer using visual components (icons, buttons, scrollbars, etc).
- These visual components are often referred to as widgets.
- A GUI also provides libraries which allow the development of new software with these visual components.
- Before GUIs were distributed with operating system, interaction with the OS was often achieved through a CLI (command-line interface).

Compilers

- A piece of software that translate instructions written in one language to another language.
  - A C compiler transforms C code into machine code.
  - A Java compiler transforms Java code into Java byte code.
- A compiler typically translates from a high-level language to a lower-level language.
- When a compiler produces machine code, its produces code for a specific platform.
  - In other words, C code compiled for Linux will not run on Windows
  - (and vice-versa).

A little bit of history ...

- The history of Unix begins in a failed operating system by AT&T Bell Laboratories called Multics.
- Ken Thompson who was working on this project, wrote a games called *Space Travel*.
- When the project was canceled, he decided to port the game to the PDP-7 computer.
- He wrote Unix as an operating system to make it easier to port the game.

Open vs Closed Operating Systems

- An OS is closed (or proprietary) when it owned by a single company.
  - It is often designed to work on a single kind of hardware.
- An OS it open (or non proprietary) if no single company is responsible for its development.
  - Since many people can work on the code, it is often available to different hardware platforms.

Open or Closed

Open

- FreeDos
- Linux
- FreeBSD
- OpenBSD

Closed

- MS Dos
- MS Windows
- HP-UX
- AIX

Types of Unix

- System V UNIX : Operating Systems based on the original AT&T UNIX code fit in this category. These include most commercial UNIX distribution.
  - AIX, Iris, Solaris, UnixWare, etc.
- BSD UNIX : These Operating Systems are based on the Berkeley Software Distribution (BSD) version of UNIX.
  - FreeBSD, OpenBSD, NetBSD and MacOS X.
- UNIX-like systems : Several Operating Systems behave like UNIX, but are not based on the original AT&T code.
  - Linux, Hurd, Minix

UNIX Standards

- Given the numerous different flavors of UNIX its available, its not surprising that there was an important need for standards.
- For vendors, it was very difficult to make their application function on different Operating Systems.
- Thus, the POSIX (Portable Operating System Interface) were established.