CSCI 2132 – Software Development

Introduction

January 9, 2017

Dalhousie University

Meng He

Learning Goal 1: “Program in the Large”

- How to write large computer programs
  - Software systems consisting of a large number of modules (smaller programs)
  - Modules are often written by different programmers
- Specific Techniques
  - Software development processes
  - Source code management
  - Software testing and debugging

Learning Goal 2: Low-Level Programming

- Understand how computer systems work in the low level
  - High level: Closer to users; high-level abstraction
  - Low level: Closer to hardware
- This supports goal 1
  - Would you like to have someone design a car without understanding how a car works?
  - Provide examples of abstractions

Why C?

- A low-level programming language
  - Closer to assembly language / machine code than Java and other high-level programming languages
- The freedom it gives to its programmers motivated discussions of software engineering principles
- Widely used
  - Systems written in C: Unix, Linux, …
  - C-based programming languages: C++, PHP, Java, C#

Why UNIX?

- Does not hide operating system operations
- Linux is open-source
- Widely used
  - Servers

One Sentence Summary

- This course helps you become an effective software developer

Course Organization

- Instructor: Meng He
- Office: 315
- Office hours
  - Mondays and Wednesday: 1-2pm (starting from Jan. 16)

Textbooks

- K. N. King, *C Programming: A Modern Approach*, 2nd Edition, W. W. Norton & Company, 2008.
- Graham Glass and King Ables, *UNIX for Programmers and Users*, 3rd Edition, Prentice Hall, 2003.

Coursework and Grading

- Seven Assignments (A)
  - Best six out of seven assignments will be used
- Midterm I (M1)
  - February 8
- Midterm II (M2)
  - March 10
- Final exam (F)

- Final Grade =
  A * 30% + max(M1*10%+M2*10%+F*50%, F*70%)

Lectures

- A few lectures will be given using Powerpoint
  - Slides will be made available online
- Long examples (programs)
  - Will be projected
  - Code will be available electronically: few comments, with blanks
  - Let's do the fill-in-the-blank questions in class (take notes of the answers)
  - Notes about design and some comments will be given
  - After class, you are advised to fill in the blanks and add comments, run them on bluenose, and print them to study them
- For all other content
  - One set of notes will be made available online

Online and In-Class Course Notes

- Online notes are like articles, so they look differently from the notes taken in class
- In-class notes focus more on main ideas of the lectures
- Online notes also include most of the instructor's verbal explanations

Suggestions on How to Make Use of Online Notes (I)

- If you feel that the pace of this course is not too fast and you have enough time to think about the questions that the instructor asks in class
  - Take notes in class
    - This gives you extra practice
    - It may be useful to have a set of notes that serve as summaries of main ideas
  - Read the online notes after class to confirm your understanding

Suggestions on How to Make Use of Online Notes (II)

- Otherwise
  - Attend lectures without taking notes (still take notes of the blanks in source code)
  - Instead, focus on:
    - Understanding the main points that the instructor spends time explaining
    - Thinking about the questions that the instructor asks in class and participating
  - Print and study the online notes after class
- You can also come up with your own strategy
- But do not stop coming to lectures just because notes are available

Assignments

- Dates
  - Posted on Wednesdays
  - Due at 3pm on due dates
  - Designed to be weekly assignments
  - You have more time for some assignments
  - The exact dates are on the course information sheet
- Grace period and late policy
  - No late assignment will be accepted
  - A grace period of 30 minutes is granted to each assignment
- Difficulty ratings
  - 3 bronze, 2 silver, 2 gold
  - A small number of bonus marks are available for gold assignments

Marking Schemes of Programming Assignments

- Programming assignments will be evaluated for
  - Correctness
  - Design
  - Documentation
- Correctness
  - This will be evaluated using an automatic testing program
  - Similar to client evaluation of software product
  - Failing to pass more than half of the test cases will affect your design / documentation marks as well, since a significant portion of the program is missing / incorrect
- Disclaimer: This does NOT apply to coding questions in exams

What to do When your Program is Incorrect?

- Do:
  - Debug!
  - Try to make your program run for at least some of the simple cases if you run out of time
  - You will learn a lot from this debugging process
  - This is how your software products will be evaluated by your clients in the future
- Do not:
  - Keep writing your program without testing it
  - These are not written assignments!
  - You will learn little by simply keep writing code
  - Normally you will get 0 if your program does not generate correct output for any test case

Lab Work

- Nine mandatory labs
  - Starting from this week
  - Course materials that are more suitable for lab work than classroom learning
  - Help to get ready for some assignments
  - The last lab will give you practice on course materials that are not covered by assignments because of academic deadlines
- No labs during the study break
- Other labs
  - Assignments and practice questions

Programming Environment: Labs

- In the lab
  - SSH from Mac/PC
  - Server: bluenose.cs.dal.ca
- At home
  - SSH from Mac/PC/Linux
  - Work on Linux PC directly: All programs will be tested at bluenose.cs.dal.ca

Website

- http://web.cs.dal.ca/~mhe/csci2132/
- Site map
  - Home/Announcements
  - Course Information
  - Lectures
  - Assignments
  - Labs
  - Practice questions
  - Exams
- Check announcements regularly

Last, but Very Important

- Historically, many students fail this course

- Why?
  - First-year courses here tend to be easy, (perhaps) in order to foster interest in computer science
  - But computer science is not easy: there is a reason why the typical pay of a software engineer is higher than most others'
  - This is one of the first courses that require students to spend a lot of time studying

Fortunately…

- In recent years, the percentage of students who failed this course was not very high
  - Students worked hard
  - Instructor made it very clear what to study (sample exams, practice questions, suggestions on exam preparation, etc)
- Conclusions
  - This course requires students to spend a lot of time
  - Working hard in this course will be rewarding
- Suggestions
  - Start working on assignments early (remember the client evaluation of software products)
  - Study after each lecture and practice
  - Ask questions if there is anything that you do not understand

Questions?