CSCI 2132
Software Development

Lecture 1:

Course Introduction

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Course Description: What is this course about?

- Introduction to intermediate programming and software development techniques
- Command-Line Interface, Procedural Language (C), a UNIX-style operating system (Linux)
- Tools and techniques: source code management and version control, build tools (make), software testing, debugging, scripting, and other techniques useful for software development

CSCI 2132: Software Development

Time:	Mondays, Wednesdays, and Fridays 12:35–13:25
Location:	Chemistry 125
Labs:	B01 Thu 08:35–09:55, Goldberg CS 143 (TLab 2) B02 Thu 08:35–09:55, Goldberg CS 133 (TLab 1) B03 Thu 08:35–09:55, LSC-Common-Area 220 B04 Thu 10:05–11:25, Goldberg CS 143 (TLab 2) B05 Thu 08:35–09:55, Goldberg CS 133 (TLab 1) B06 Thu 08:35–09:55, Goldberg CS 143 (TLab 2)
Instructor:	Vlado Keselj (Vlado Kešelj, pron.≈ Vlado Keshel) office: CS 432, email: vlado@dnlp.ca, phone 902-494-2893
URL:	`http://web.cs.dal.ca/ vlado/csci2132`
E-mail list:	csci2132@lists.dnlp.ca

Some Important Dates

- *More information on the course calendar page*
- Term starts: Tue Sep 4, 2018
- Last day to add classes: Tue Sep 18, 2018
- Midterm Exam I: Thu Sep 27, 2018
- Last day to drop class without “W”: Mon Oct 1, 2018
- No class, Thanksgiving: Mon Oct 8, 2018
- Last day to drop class with “W”: Tue Oct 30, 2018
- Midterm Exam II: Thu Nov 8, 2018
- No class, in lieu of Remembrance Day: Mon Nov 12, 2018
- Fall Study Break (no classes): Nov 12–16, 2018
- Term ends: Tue Dec 4, 2017 (Monday classes held)
- Final Exam: TBA, it will be a 3h exam in the period of Dec 6 to 16, 2018

Evaluation Criteria

- Assignments (30%)
  - Tentatively 7-10 assignments, best $n-1$ used for grading if $n > 6$
  - **Late assignments will not be accepted.**
  - Assignments will be submitted electronically; exceptions possible
  - Will likely include two practicums during lab time with requirement to solve at least one problem
- Midterm Exams (20%)
  - Two midterms, during class time
- Final Exam (50%)
  - Scheduled by the university.
  - Will cover all material in the course.
  - Midterms may be ignored if better mark is obtained by ignoring them and counting 70% for the Final Exam

Lectures

- Slides and notes will be available online
- Longer examples (programs)
  - Code will be available electronically: few comments, with “blank” part
  - Will do the fill-in-the blank questions in class, and it is advised that you 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

Midterm and Final Exam Requirements

- Photo ID is required.
- Closed book. One single prepared sheet with up to two pages allowed (“cheat sheet”).
- No calculators, cell phones, notes, dictionaries, and other electronic of paper aids allowed.

Marking Schema 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
  - Your program must compile and pass at least the test case given in the assignment
- 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

Lab Work

- Labs are mandatory
- Course materials that are more suitable for lab work than classroom learning
- Helps to get ready for some assignments
- The labs will likely include some course material that is not covered in lectures or assignments
- Some labs may be canceled, but you can still use the labs for your own practice

Programming Environment: Labs

- In the lab
  - SSH from Mac/Windows (use putty on Windows)
  - Server: bluenose.cs.dal.ca
- At home
  - SSH from Mac/Win/Linux
  - Work on Linux PC directly: All programs will be tested at bluenose.cs.dal.ca
  - You can also use VirtualBox on your own computer

Academic Integrity Policy

- Please read the given handout (also available at the course web site)
- Suspected cases of plagiarism are referred to Academic Integrity Officers, and may lead to serious consequences
- Plagiarism is defined as “the presentation of the work of another author in such a way as to give one’s reader reason to think it to be one’s own”
- Fully reference sources in your assignments and reports
- You can look at other code, but do not cut-and-paste!
- Discussing assignments verbally is likely not an issue, but do not discuss it in writing or typing

Dalhousie Culture of Respect

- We believe that inclusiveness is fundamental to education and learning.
- Every person has a right to be respected and safe.
- Misogyny and disrespectful behaviour on campus, wider community, and social media is not acceptable. We stand for equality and hold ourselves to a higher standard.
- Take an active role:
  - Be ready: do not remain silent
  - Identify the behaviour, avoid labeling, name-calling or blame
  - Appeal to principles, particularly with friends, co-workers or similar
  - Set limits
  - Find an ally and be an ally, lead by example
  - Be vigilant

Required Texts and Resources

- *C Programming: A Modern Approach,* by K. N. King, W. W. Norton & Company, 2008.
- *UNIX for Programmers and Users,* by Graham Glass and King Ables, Prentice Hall, 2003.
- **Recommended Reading**
- *Unix and Linux System Administration Handbook,* by Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, edition 4th Edition, Pearson Education, 2010.
- *The C Programming Language,* by Brian W. Kerninghan and Dennis M. Ritchie, edition 2, Prentice Hall Software Series, 1988.

Course Prerequisite

- CSCI 1101 or suitable prior programming experience

Main Learning Objectives

- One sentence summary:
  - This course should help you become an effective software developer
- Divided into two learning goals:

1. Programming “in the Large”
2. Low-level Programming

Goal 1: Programming 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

Goal 2: Low-Level Programming

- Understand how computer systems work at 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?
  - Complex systems are frequently built from a low abstraction level

Why Unix-style system?

- What do we know about UNIX, Linux and similar?

Why Unix-style system?

- UNIX was the first popular multi-user OS that set a **standard,** which is stable and widely used
- Powerful Command-Line Interface **(CLI)**, corresponding to the sequential nature of computing
- Many **utilities**, that became well-known, standard tools
- **Philosophy** of elegant and modular solutions
- It has wide and significant **use** in practice: servers, Linux, BSD (MacOS), Android, etc.

Open Unix-style Model

- Does not hide Operating System operations
- Provides all the basic low-level abstractions that are used by modern Operating Systems:
  1. Text-based interface
  2. Files
  3. Processes
  4. Pipes
  5. Virtual memory (Process Isolation)

Why C?

- Widely used and portable, and still very close to machine code (i.e., assembly language)
- Efficient and gives much control to programmer
- Compiled, runs directly on the system (no VM layer)
- Does not hide the system, and allows fine-grained system manipulation
- Forces the programmer to think about many low-level issues
- Emphasizes the notion of sequential execution
- “Lingua franca” of programming world

Historic Importance of C

- Relatively old and small language, which is still very much used without significant changes
- No close alternative
- It had a major influence on a majority of modern languages: C++, PHP, Java, C#, Perl, etc.
- C and C++ are still dominant languages in large software system development (e.g., `http://www.lextrait.com/Vincent/implementations.html`)

Tentative List of Course Topics

Course Introduction

1. Fundamentals of Unix-style Operating Systems
   - History
   - Basic commands and utilities
   - Structure (files, directories, processes, . . .)
2. C Programming Language and Software Development
   - Introduction to C
   - Software development life cycle
3. Program Organization and Dynamic Memory Allocation
   - Writing large programs, make
   - Pointers and dynamic memory allocation
4. Shell Scripting and Control Version Systems