CSCI 2132: Software Development

Introduction

Norbert Zeh

Faculty of Computer Science
Dalhousie University

Winter 2019

This Course in a Nutshell

This course will be your first step towards becoming a better software developer.

Unix

- Shells
- Command-line tools

C programming

- Procedural
- Low-level
- Close to hardware

Software development

- Testing
- Debugging
- Source code management
- Software development methodologies

Programming in the Large

(Learning Objective 1)

Challenges:

- Software systems composed of multiple modules (parts)
- Modules often written by different users

Techniques:

- Software development process/methodologies
- Software testing and debugging
- Source code management

Low-Level Programming

(Learning Objective 2)

Understand how computer systems work "under the hood":

- High-level programming:
  - High level of abstraction
  - Close to user (programmer)
- Low-level programming:
  - Close to hardware

This supports learning objective 1:

- Would you like someone to design a car without knowing how cars work?
- Provide examples of abstractions
- Understand the runtime cost of abstractions

Why Unix?

- First popular multi-user OS that set a **standard**
- **Stable**
- **Powerful command-line interface** (CLI):
  Command line = GUI + IDE for power users
- Many **utilities**, well known, standard tools
- **Philosophy** of **elegant** and **modular** solutions
- **Widely used** on servers, desktops, and mobile devices
  (UNIX, BSD, Linux, macOS, Android, ...)

Open Unix-Style Model

- Does not hide operating system operations
- Provides all low-level abstractions used in modern operating systems:
  - Text-based interface
  - Files
  - Processes
  - Pipes
  - Virtual memory (process isolation)

Why C?

- **Widely used:**
  - Systems written in C: UNIX, Linux, ...
  - Languages influenced by C: C++, PHP, Java, C#, Rust, ...

- **Low-level programming language:**
  - Close to machine, gives programmer fine-grained control
  - No garbage collection, no virtual machine, compiled
  - 0-overhead principle
  - Forces programmer to think about low-level issues

- **“Lingua franca” of programming world:**
  - Interface between different programming languages
    often uses C-style calling conventions

Instructor

Name: Norbert Zeh

Email: nzeh@cs.dal.ca

Office: MC 4246

Office hours: MWF 12:00–14:00

TAs: TBD

Lectures and Labs

Lectures

MWF 3:30–4:30	McCain Auditorium 1

Labs

M 8:30–10:00	Mona Campbell 1201	B01
M 8:30–10:00	Goldberg 143	B02
M 10:00–11:30	Mona Campbell 1201	B03
M10:00–11:30	Goldberg 143	B04

Important Dates

Monday, Jan 7	Lectures start
Friday, Feb 1	Munro Day (university closed)
Monday, Feb 18	NS Heritage Day (university closed)
Feb 18–22	Study Break (no classes)
Monday, Apr 8	Last lecture

The “fun” stuff

Monday, Feb 4, 6:30–8:30 (PM)	Midterm 1
Monday, Mar 4, 6:30–8:30 (PM)	Midterm 2
TBD Apr 10–26	Final

Lectures

- Slides available online
- Longer examples (programs)
  - Code will be available electronically:
    (few comments, blank parts)
  - Blanks will be filled in in class, take notes
  - Fill in the blanks after class, run on bluenose, study the code

Exams

- Photo ID required
- Closed book
- Cheat sheet: One single sheet, front and back, is allowed
- No calculators
- No cell phone
- No notes
- No dictionaries
- No other aids (electronic or paper)

Evaluation

Assignments (30%)

- 7–10 assignments, best n–1 count
- Late assignments not accepted
- Submit electronically

Midterms (20%)

- 2 midterms, 10% each
- In the evening

Final exam (50%)

- Scheduled by university
- Covers all material covered in the course

Evaluation of Programming Assignments

Criteria:

- Correctness
- Design
- Documentation

Correctness:

- Will be evaluated using automatic testing
- Similar to client evaluation of software product
- Program must compile and pass at least the test cases given in the assignment

What to do When Your Program is Incorrect?

Do:

- Debug!
- Try to make your program work for simple cases if you run out of time.
- You will learn a lot from debugging.
- You will spend much of your time as a programmer testing and debugging.

Do not:

- Keep writing your program without testing.
- You will learn little by simply writing code without testing it.

Lab Work

- Labs are **mandatory**,
- Cover material more suitable to be explored in a lab than in lectures,
- Help you get ready for assignments,
- Will likely cover **some material not covered in lectures** or assignments.

Programming Environment: Labs

In the lab:

- `ssh` from Mac/Windows (use PuTTY on Windows)
- Server: `bluenose.cs.dal.ca`

At home:

- `ssh` from Mac/Windows/Linux
- Work directly on a Linux PC
- Run Linux in a VirtualBox

Note: All evaluation will happen on bluenose

- Make sure your code compiles and runs correctly on bluenose

Academic Integrity Policy

https://www.dal.ca/dept/university_secretariat/academic-integrity.html

- Suspected cases of plagiarism referred to Academic Integrity Officer
  - Serious consequences if found guilty
- **Plagiarism = “presentation of work of another author as your own”**
- Fully reference sources in your assignments and reports
- You can look at other code, but do not
  - Cut and paste
  - Copy verbatim (or with only cosmetic changes)
- You can discuss assignments, do not exchange notes

Dalhousie Culture of Respect

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

Textbooks

Required:

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

Recommended:

- E. Nemeth, G. Snyder, T.R. Hein, and B. Whaley. *UNIX and Linux System Administration Handook.* 4th ed. Pearson Education, 2010.
- B.W. Kerninghan and D.M. Ritchie. *The C Programming Language.* 2nd ed. Prentice Hall Software Series, 1988.