Lecture 1 —Introduction and Our C Toolkit

Prepared by Jeff Zarnett, taught by Seyed Majid Zahedi
jzarnett@uwaterloo.ca, smzahedi@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

Course Syllabus

As our first order of business, let’s go over the course syllabus.

Collaborative Course

The source material for the ECE 252 notes and slides is open-sourced via Github.

If you find an error in the notes/slides, or have an improvement, go to `https://github.com/jzarnett/ece252` and open an issue.

If you know how to use `git` and LaTeX, then you can go to the URL and submit a pull request (changes) for me to look at and incorporate!

Some Background on the Operating System

An operating system (OS) sits between the hardware and programs.

It does many different things and has many often-conflicting goals.

You might think of the OS as the “secretary” of the system.

OS: Resource Manager

The OS is responsible for resource management and allocation.

Resources like CPU time or memory space are limited.

The OS must decide how to allocate & to keep track of system resources.

In the event of conflicting requests, choose the winner.

OS: Environment Provider

The OS enables useful programs like Photoshop or Microsoft Word to run.

The OS is responsible for abstracting away the details of hardware.

Imagine Hello World had to be written differently for different hardware.

OS: Multitasking

Multiple programs means some resources are shared.
$\rightarrow$ A source of conflicts!

OS creates and enforces the rules so all can get along.

Sometimes processes want to co-operate and not compete.
The OS can help them to do so.

The OS is the backdrop to what we will do in this course: systems programming.

Systems Programming

Some examples of systems programming:

- **File Manipulation**
- **Communication**
- **Processes and Thread Management**

Systems Programming

Programming at this level is more difficult than regular programs.

It may require knowledge of the hardware, or perhaps programming facilities like debugging are limited.

Systems programs must take concurrency into account.

Ask for Help

We want to do certain operations that involve the operating system.

There are things that the operating system does not allow programs to do.

What they have to do instead is ask the operating system to do it instead.

A program is said to be concurrent if it can support two or more actions in progress at the same time.

It is parallel if it can have two or more actions executing simultaneously.

Soon enough we will spend a great deal of time examining the differences between parallelism and concurrency in the program.

Concurrency

It is already the case that many programs you use are to a greater or smaller degree concurrent.

Depending on your level of programming experience, you may have already written a concurrent program, intentionally or without knowing it.

We will learn about how to take a program and make it concurrent, as well as how to write it with concurrency in mind from the ground up.

Concurrency Problems

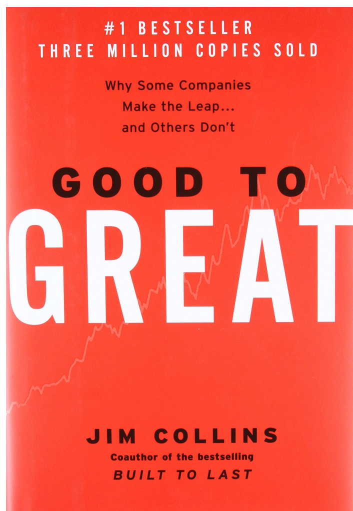
Consider a program that performs a simple calculation given some input.

If the program has a concurrency problem, then the answer could be:

1. Consistently the wrong answer every single time
2. Different on consecutive runs with the same input, or
3. Correct some of the time but incorrect some of the time.

As you can imagine, none of these options are acceptable.

But why are you here? Why does this matter?



Do you want to be great?

This course is going to be hard.

If your programming skills need work, better to start trying to catch up now.

Our C Toolkit

We will need some introduction to the conventions and tools of C:

- Functions
- Header files
- Comments
- Structures
- Type Names
- Memory Allocation, Deallocation, and Pointers
- Dereferencing, Address-Of, The Arrow
- Arrays
- Strings
- Calling Conventions & Errno
- Printing
- Constants
- `main` and its arguments
- `void*`