CMPT 125: Introduction to Computing Science and Programming II

Mo Chen
SFU Computing Science
6/1/2020
https://www.sfu.ca/~mochen

Lecture 1 Plan

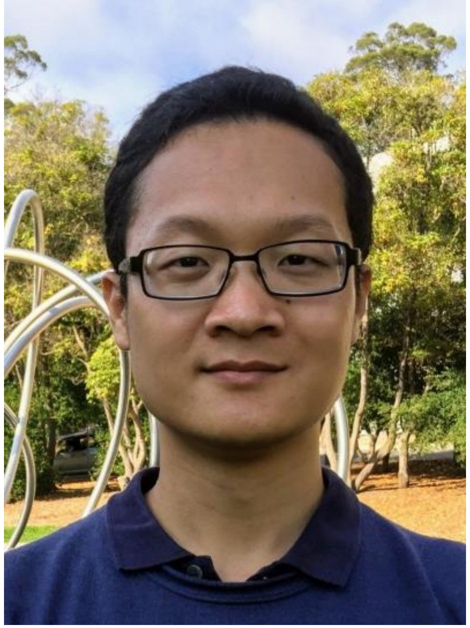
Today:

- Introductions
- CMPT 125 vs CMPT 127
- Grading scheme
- Other expectations
- Computer science review / overview
- Running a program in C

Introductions

Two instructors for two courses!

CMPT 125 and CMPT 127



Mo Chen

Anne Lavergne

About Me

- Undergraduate degree from UBC
- PhD degree from UC Berkeley
- Postdoc from Stanford
- Assistant Professor at SFU CS since 2018
  - Multi-Agent Robotic Systems Lab (https://sfumars.com)

CMPT 125 vs CMPT 127

- Co-requisite courses
  - You must take them as a pair
  - Separates theory from practice
- CMPT 125 will be focused on algorithms, computer science, analysis
  - Assigned work on paper and computer
- CMPT 127 will be focused on writing code, debugging, testing, Linux tools
  - Assigned work on computer

CMPT 125

- Lectures MWF 12:30-13:20, SWH 10081
  - Website: https://coursys.sfu.ca/2020sp-cmpt-125-d1/pages/
- Piazza: online discussion and Q&A
  - Sign up: https://piazza.com/sfu.ca/spring2020/cmpt125
  - Main page: https://piazza.com/sfu.ca/spring2020/cmpt125/home
- Office hours:
  - Before and after lectures
  - Fridays 15:30-16:30, TASC 1 8225
  - See website for TA office hours info

CMPT 125

- 9 Assignments:
  - ~Weekly (total of 20%, lowest assignment is dropped)
  - Late policy: 3 grace days; no late submissions afterwards
- In-class Midterm:
  - Wednesday, February 26 (25%)
- Final:
  - Thursday, April 19, 12:00-15:00 (55%)

Other Expectations

By the end of this course you can expect to be able to:

- write high quality code in C
- use standard command line tools in Linux
- develop algorithms to solve problems
- predict the behaviour of algorithms

Other Expectations

From CMPT 120, it’s assumed that you are proficient at the basic concepts of programming.

- Data types and conversions (integer, float, string)
- Expressions (a+b*c)
- Basic terminal input/output (raw_input() and print)
- Libraries (import from modules)
- Conditionals (if-elif-else)
- Definite loops (for) and indefinite loops (while)
- Functions and parameter passing
- The [develop → test → debug] cycle

Other Expectations

You are not expected to know the C syntax for these concepts, only that you know the concepts

- Over the first few weeks, you will learn how they are expressed in C

Other Expectations

Our expectations of you:

- 10 hours per week per course
    - standard workload for SFU courses
- CMPT 125 = 3 hours lecture + 7 hours reading / studying / solving assignment problems

Other Expectations

RESPECT

Theme: Do not interfere with the learning of others.

- show up to class on time
- no talking during class [about non lecture-related material]
- no texting / Facebook / youtube in the e-free zone - sit in the back row of class if you ***must*** do this
- complete / submit your OWN work == be academically honest

Bottom line: Do not interfere with the learning of others.

Course Objectives / Outline Summary

- Two courses; two co-instructors
- Lecture course is computer science focused
- Lab course is computer programming based
- Both courses are fundamentals - put in the time and your future work will be easier
- Respect your classmates, both inside and outside of the classroom / lab.

Any questions?

Computer Science Review / Overview

What is Computer Science?

[From CMPT 120]

- The study of algorithms, their formal and mathematical properties, their hardware realizations, their linguistic realizations, and their applications.

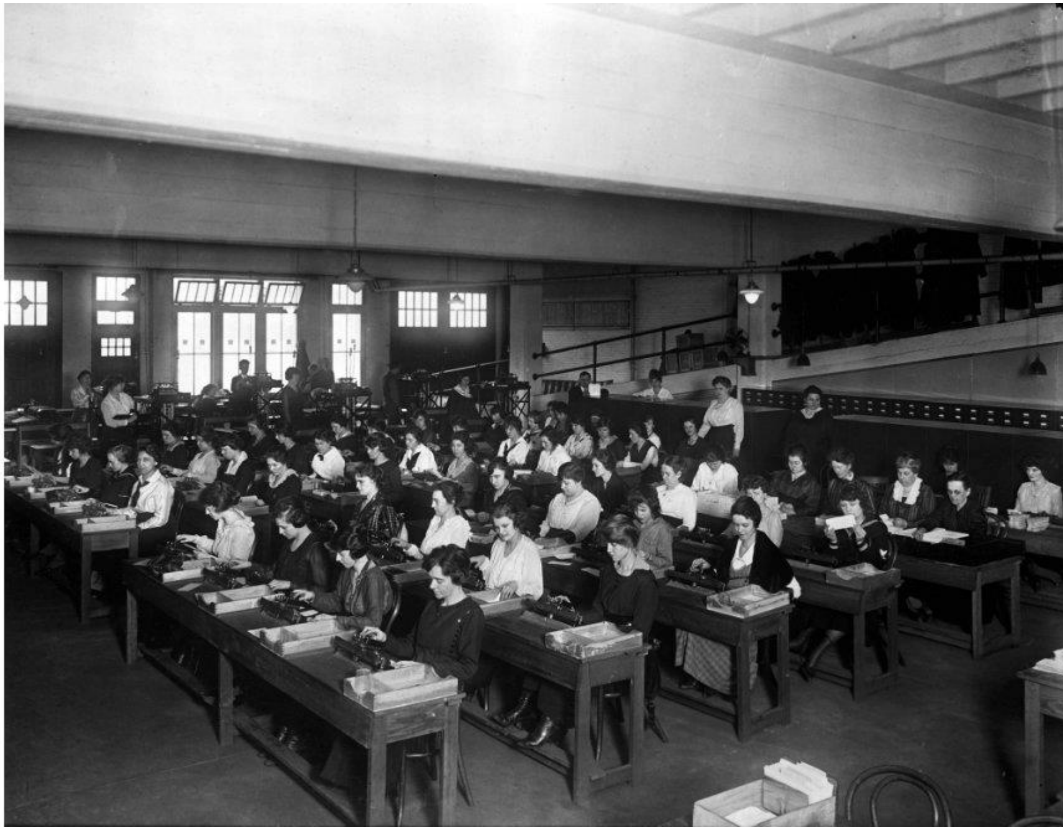
[From real life]

- The study of what computers can and cannot do.

Computer Science Review / Overview

The very first computers were utilized to perform pure calculation: tables for sin(x), cos(x), log(x)

- Human calculators replaced by automation!
- “Calculator” and “Computer” used to be job titles!



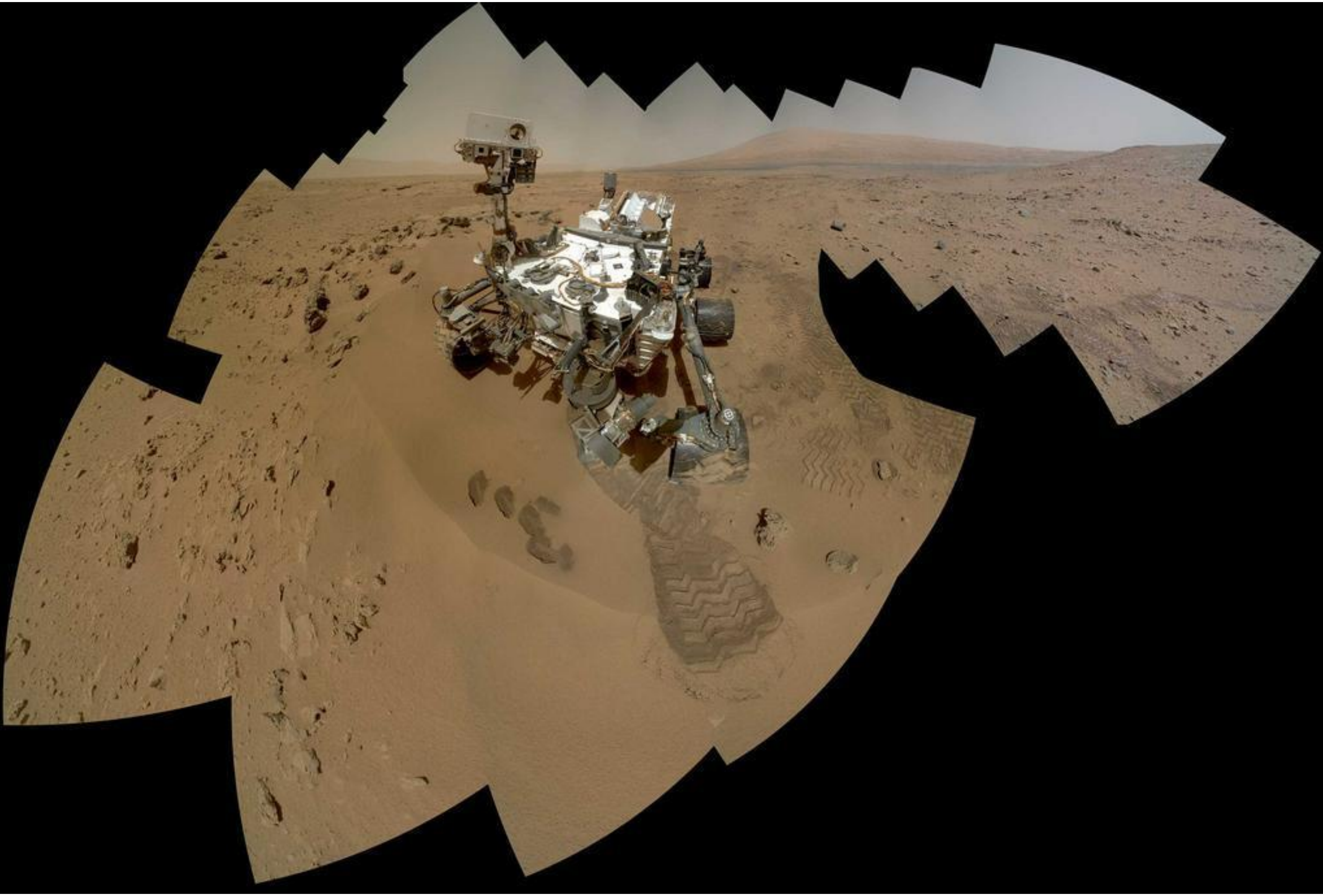
Computer Science Review / Overview

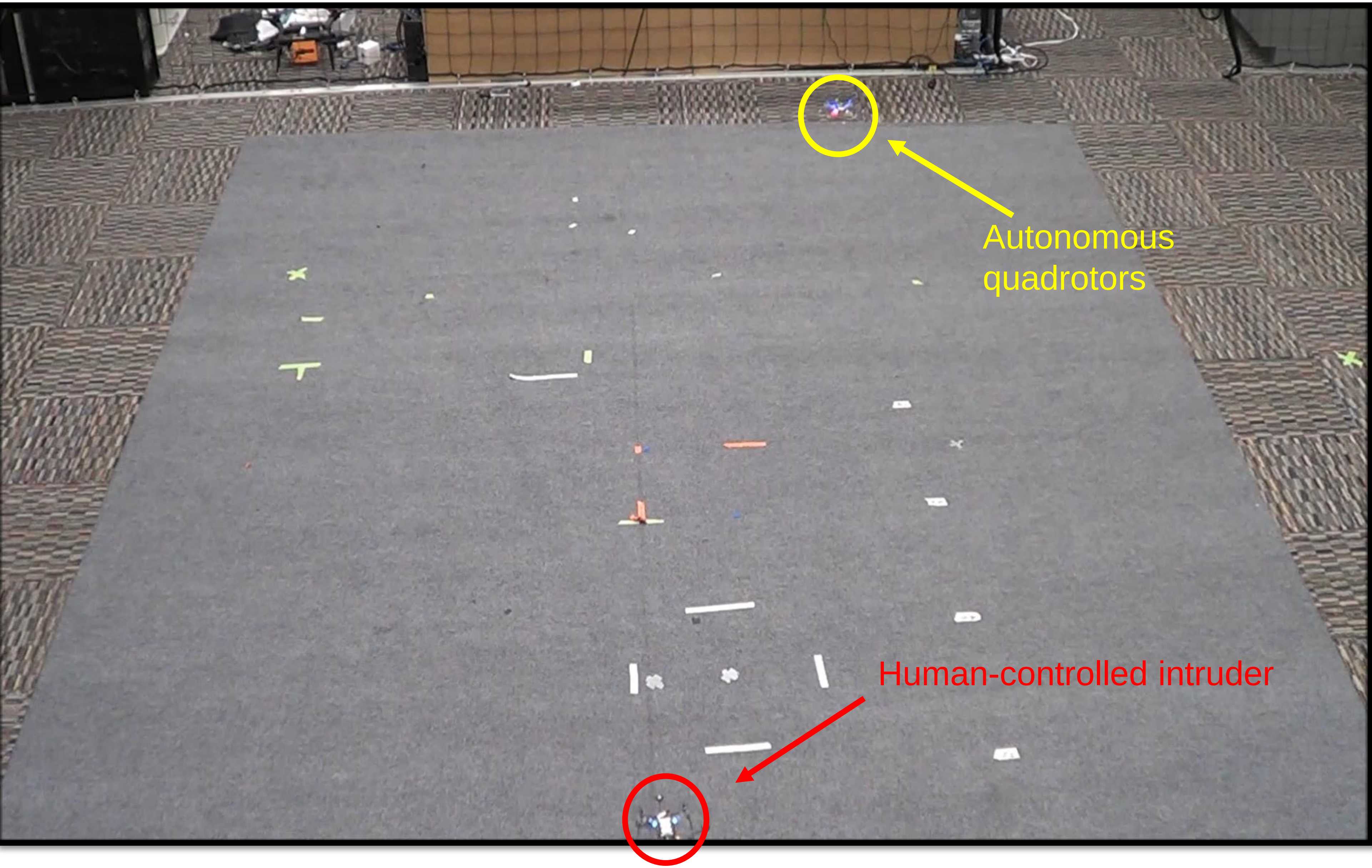
Computing is applied everywhere

- big - mainframes, supercomputers
- medium - desktop, laptop, tablet
- small - smartphones, cars, microwaves

Automating more and more of our society

XPS
SONY
PS4
SAMSUNG
DUOS
12:45
at&t
Start
XBOX
tv
DELL




Autonomous quadrotors
Human-controlled intruder

Computer Science Review / Overview

Algorithms are the core component.

- Definition: An *algorithm* is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining some required output for any valid input in a finite amount of time.
- You communicate algorithms to computers using a programming language.

Computer Science Review / Overview

programmingLanguageForCMPT125And127 = C;

- Well, it will be mostly just C, but seasoned with some of the elements of C++.

Why C?

- Because it is everywhere.

Running a Program in C

3 Steps:

1. Edit your program.
   - Use “`gedit`”. (or any other text editing program)
   - Save in a `.c` file.
2. Compile your program.
   - Use “`gcc program.c`” . . .
   - . . . to generate “`a.out`”.
3. Run your program.
   - Use “`./a.out`”.

Running a Program in C

Step 1: “`gedit`”

(screenshot of empty window)

or . . .

Running a Program in C

Step 1: “`gedit program.c`”

(it’s still a blank window, but it saves to the “right” location)

Running a Program in C

`gedit`

- a simple editor, like Notepad (Windows) or TextEdit (Mac)
- does text highlighting for C syntax

Running a Program in C

```
#include <stdio.h>
```

“`#include`” in C is like “`import`” in Python

Running a Program in C

```
#include <stdio.h>

int main ( ) {

}
```

This is your main function - it is always where your program starts its execution.

Running a Program in C

```
#include <stdio.h>

int main ( ) {

}
```

Curly braces `{ }` denote a block of code.

(Like block indentation does for Python.)

Running a Program in C

```
#include <stdio.h>

int main ( ) {
    printf("Hello world\n");
}
```

- `printf(...)` is your output function.
- All statements end with a semicolon “`;`”.
- Newlines are not automatic: use “`\n`”.

Running a Program in C

Save your program as a `.c` file

Open a console window to get to the command prompt, and run the C *compiler*

```
>$ gcc program.c
>$
```

If successful, creates an executable program called “`a.out`”.

Running a Program in C

You are finally ready to run your program!

Type “`./a.out`” as your next command

```
>$ gcc program.c
>$ ./a.out
Hello world!
>$
```

Acknowledgement

The slides of this course are the work of Brad Bart (with minor modifications)