APS 105S — Computer Fundamentals

Introduction

Baochun Li, Professor
Department of Electrical and Computer Engineering
University of Toronto

Welcome!

- Welcome to APS 105S — Computer Fundamentals
  - An introduction to software

Who am I?

- Professor, Computer Engineering Group, ECE
- Research Areas: cloud computing systems, distributed systems, networking

My background

- Been teaching this course since 2001 (F2001-F2005, W2008, F2008-F2012, F2014-F2015, W2017-W2019)
- Education
  - 1990 — 1995: B.Engr.,Tsinghua University, Beijing, China
  - 1995 — 1997: M.S., University of Illinois at Urbana-Champaign, US
  - 1997 — 2000: Ph.D., University of Illinois at Urbana-Champaign, US

My career at the University of Toronto

- 2000: Assistant Professor
- 2005: Associate Professor
- 2008: Full Professor
- 2015: IEEE Fellow

What is this course about?

It’s about computer programming

Why learn programming?

- Computer programs are used to build software
- Software is pretty cool!
  - Facebook, Twitter, Google, smartphone apps — all built with software
  - 60% of all trades in the New York Stock Exchange are placed by software
  - Many engineering jobs, whether civil, electrical or mechanical, involve simulations, built with software

The difference between the best worker on computer hardware and the average may be 2 to 1, if you're lucky. With automobiles, maybe 2 to 1. But in software, it's at least 25 to 1. The difference between the average programmer and a great one is at least that.

— Steve Jobs, 1995

We'll learn programming in the C programming language

Why learn programming in C?

- There are other newer languages — Python, Javascript, Java, Swift
- Benefits of C
  - Compiled language, rather than interpreted
  - Closer to the hardware, and thus easier to understand what is happening
- C and its variants (C++) are still very commonly used
  - to build high performance applications (such as git) and operating system kernels
- C-like syntax is found in many languages, such as Java and Swift

Recent survey of language popularity

Worldwide, Jan 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %
6		PHP	6.06 %	+0.0 %
7		R	4.19 %	+0.3 %
8		Objective-C	2.27 %	-1.4 %
9		Swift	1.91 %	-0.2 %
10		TypeScript	1.74 %	-0.0 %

https://pypl.github.io/PYPL.html

Drawbacks of the C language

- Some complexities that are hard for new programmers to learn
- Sometimes considered too low-level: to complete the same task, more lines of code need to be written

Course Syllabus

- Please read the course syllabus
  - available for download on the course website
  - very useful and detailed information there
- Let’s get started and discuss some details today

Overview of the course

- Three instructors, four lecture sections
- Grading: 30% labs, 30% midterm, 40% final exam
- Course website: http://q.utoronto.ca, look for APS 105
  - All course handouts, including the syllabus, course schedule, and labs, are available in the course website
  - Announcements are in the “Announcements” section
  - Syllabus and teaching schedule are available on the front page

Piazza online discussion forum

- To signup:
  - piazza.com/utoronto.ca/winter2022/aps105
- To visit:
  - piazza.com/utoronto.ca/winter2022/aps105/home
- First check the forum to see if your question has already been answered
- Good idea to take a look from time to time

Labs

- Nine labs in the course (all marked)
- SF 1012 and 1013 (Sanford Fleming)
- Handout available for download two weeks before
- Submit your lab by the end of your PRA section in the week it is due
- Marked by the **automarker** program on **examify.ca**
  - Some test cases are given to you, others are not
  - Your output will be compared with standard output
  - Must output exactly as prescribed

Tutorials

- 8 tutorial sections — you have to attend yours
- A total of 10 tutorials in this course — see schedule
- Led by a tutorial TA
- Discuss material and solve problems covered by previous lectures
- First tutorial: in the week of January 17, 2022

Plenary lectures

- Tuesdays, 5 — 6 p.m., online and/or MS 2158
- Optional attendance
- To prepare students before a lab is due, and to introduce closely related material beyond the lectures
- Starting on January 18, 2022, Tuesday

No Textbook — Recommended References

- The C Programming Language (Brian W. Kernighan, Dennis Ritchie), Second Edition
- Programming in C (Stephen G. Kochan)
- Lots of online tutorials and references

Academic Integrity

- Do not copy lab solutions from others
- Do not provide portions of your own solutions to others
- Do not post questions online or anywhere in the “cloud” (instant messaging, social networking sites, forums, web sites) for others to solve
- Failure to adhere to these rules is considered cheating by both parties involved
- Cheating detection programs in place
  - Penalties are imposed by the Faculty
  - Not worth it

Lots of ways of getting help

- Instructors: Office Hours
  - My office hour is Mondays 10:15 — 11:15 a.m., Zoom meeting link available in the course syllabus
- Labs: Help from your assigned TA online or in person
- Tutorials: Ask questions to the tutorial TA
- Plenary seminars: designed to help those who prefer a slower pace — ask questions to the lecturer

Now let’s get started

Computer Fundamentals

- Central Processing Unit (CPU)
  - Control unit and arithmetic-logic unit
  - Examples: Intel Core i3 and i5, ARM CPUs
- Main Memory
  - Indexed by addresses
- Input/Output devices — What are the examples?

Software:
Operating System and Applications

What does the Operating System do?

- Manage hardware resources
  - CPU
  - Main memory
  - Non-volatile storage (Hard drives or Solid-State Drives)
- Provide an interface to the applications
  - So that developing these applications would be easier

Typical Operating Systems

- UNIX Variants —
  - macOS (a UNIX variant)
  - Linux (another UNIX variant, used in the ECF labs)
  - Many many others (such as iOS and Android)
- Microsoft Windows
  - Based on the Windows NT kernel (the core of the OS) back in the late 1990s

Program Development Cycle: Edit — Compile — Run — Debug

Command-line tools in UNIX: Live Demo