CSE 374
Programming Concepts & Tools

Hal Perkins

Winter 2017

Lecture *n* – Course Wrapup

Reminders

- Please fill out course evaluations before they close this weekend

- Final exam: Wed. 3/15, 2:30-4:20, here
  - Topics: mostly C/C++ and tools (make, git, etc.), but likely also some shell scripting, etc. No heavy-duty regexp/sed or similar things
  - Last minute review Q&A Tue. 3/14, 4:30, MOR 220. Might be 10 min. late depending on how fast people can get there from previous exams

- So let's take a look back at what we've done this quarter…

But first...

A huge thanks to the folks who made the course work:

10 weeks ago….

- We have 10 weeks to move to a level well above novice programmer:
  - Command-line tools/scripts to automate tasks
  - C programming (lower level than Java; higher than assembly)
  - Tools for programming
- That's a lot!
- Get used to exposure, not exhaustive investigation
  - This is not intro programming anymore

What is CSE 374?

- Something of a "laundry list of everything else", but…

  There is an amorphous set of things computer scientists know about and novice programmers don't. Knowing them empowers you in computing, lessens the "friction" of learning in other classes, and makes you a mature programmer.

- The goal is to give you a sense of what's out there and what you can expect – and how you can learn more later when you need to

A few General Areas

1. The command line
   - Text-based manipulation of the computing environment
   - Automating (scripting) this manipulation
   - Using powerful *utility* programs

- Let the computer do what it's good at so you don't have to!
- We'll use Linux (an operating system) and bash (a *shell*) – but the concepts are not tied to these
- Idea: Knowing the name of what "ought to exist"
- Idea: Programming in a language designed for interaction

A few General Areas

2. C (and a little C++)
   - "The" programming language for operating systems, networking, embedded devices, …
   - Manual resource management
   - Trust the programmer: a "correct" C implementation will run a program with an array-bounds error and whatever happens, happens
   - A "lower level" view of programming: all code and data sits together in a "big array of bits"

- Idea: Parts look like Java – don't let that deceive you!
- Idea: Learn to think before you write, and test often

A few General Areas

3. Programming tools – so far you have written programs and run them. There are programs for programming you should know about:
   - Compilers (vs interpreters)
   - Debuggers
   - Linkers
   - Recompilation managers
   - Version-control systems
   - Profilers
   - …

A few General Areas

4. Software development concepts – what do you need to know to write a million lines of code?
   - Testing strategies
   - Team-programming concepts
   - Software specifications and their limits
   - …

A few General Areas

5. Basics of concurrency – what happens when more than one thing can happen at once in a program?
   - Brand-new kinds of bugs (e.g., races)
   - Approaches to synchronization
   - And it matters – most computers you can buy have (at least) 2 processors
     - How do we run enough stuff concurrently to keep all the processors busy?

Perspective

"There is more to programming than Java methods"

"There is more to software development than programming"

"There is more to computer science than software development"

You've got a lot more in your toolkit than you had 10 weeks ago

Have fun building great things!