Introduction to C

CS 2022: Introduction to C

Instructor: Hussam Abu-Libdeh

(based on slides by Saikat Guha)

Fall 2011, Lecture 1
Instructor: Hussam Abu-Libdeh, 4139 Upson
Email: hussam@cs.cornell.edu
Meeting times:
Mondays 12:20 – 1:10pm  245 Olin Hall
Wednesdays 12:20 – 1:10pm  245 Olin Hall
Fridays 8:00 – 8:50am  361 Upson Hall
Duration: 8/24 – 9/23
Office Hours: To Be Determined
http://cms.csuglab.cornell.edu/
Pre-requisites

- Basic programming knowledge (variables, functions, loops)
- Lots of composure
  - Your programs won’t compile
  - Your programs won’t run
  - Your programs will crash
  - You’ll have no idea what happened
  - … but at least it’ll happen fast!
History of C

- Writing code in an assembler gets real old real fast
  - Really low level (no loops, functions, if-then-else)
  - Not portable (different for each architecture)
- BCPL (by Martin Richards): Grandparent of C
  - Close to the machine
  - Procedures, Expressions, Statements, Pointers, . . .
- B (by Ken Thompson): Parent of C
  - Simplified BCPL
  - Some types (int, char)
History of C

- C (by Kernighan and Ritchie)
  - Much faster than B
  - Arrays, Structures, more types
- Standardization
- Portability enhanced
- Parent of Objective C, Concurrent C, C*, C++
When to use C

- Working close to hardware
  - Operating System
  - Device Drivers
- Need to violate type-safety
  - Pack and unpack bytes
  - Inline assembly
- Cannot tolerate overheads
  - No garbage collector
  - No array bounds check
  - No memory initialization
  - No exceptions
When not to use C

Use JAVA or C# for . . .
- Quick prototyping
  - Python or Ruby are even better here
- Compile-once Run-Everywhere
- Reliability is critical, and performance is secondary
  - C can be very reliable, but requires tremendous programmer discipline
  - For many programs, JAVA can match C performance, but not always