CS262 Lecture 01 Introduction

Zoran Duric
Department of Computer Science

GEORGE MASON UNIVERSITY

Introduction to Low-level Programming

- instructor: Zoran Duric
- zduric@cs.gmu.edu
- Office: ENGR 4443
- Office hours: MTW 3-4pm
- We will communicate through Piazza
  - https://piazza.com/gmu/fall2014/cs262/home
- TA's office hours will be posted on Piazza
- You will post your assignments on Blackboard
- Assignment will be graded on Blackboard

Introduction to Low-level Programming

- Monday/Wednesday 1:30 am - 2:45 pm
- Meets in Sandbridge Hall 107
- Full semester, meets two times per week
- Course webpage:
  - http://cs.gmu.edu/syllabus/syllabi-fall14/CS262All%20Instructors.html

Prerequisites

- C or better in
  - CS 211 (OOP) or

- No exceptions

Course Scopes

- Most high-level programming languages insulate the programmer from the realities of the hardware on which the programs will run

- Examples are:
  - memory management
  - file system management
  - process management
  - hardware signals

Course Scopes

- C is the exception since it was originally designed to implement the Unix operating system

- C offers the programmer direct access to much of the underlying hardware and, for programs running under Unix, direct access to operating system services

Course Scopes

- For these reasons C remains the language of choice for systems programming.
  - What are other reasons?
  - What are your reasons?

Course Scopes

- This is a (short) course on "low-level" programming using C
- We will learn C with heavy emphasis on pointer operations, i.e.,
  - how to allocate, manipulate, free memory without crashing your code

Course Outcomes

- Be able to implement, **test** and **debug** a designed solution to a problem in a low-level programming language, specifically the C programming language.

- Demonstrate a good understanding of C language constructs such as pointers, dynamic memory management, and address arithmetic.

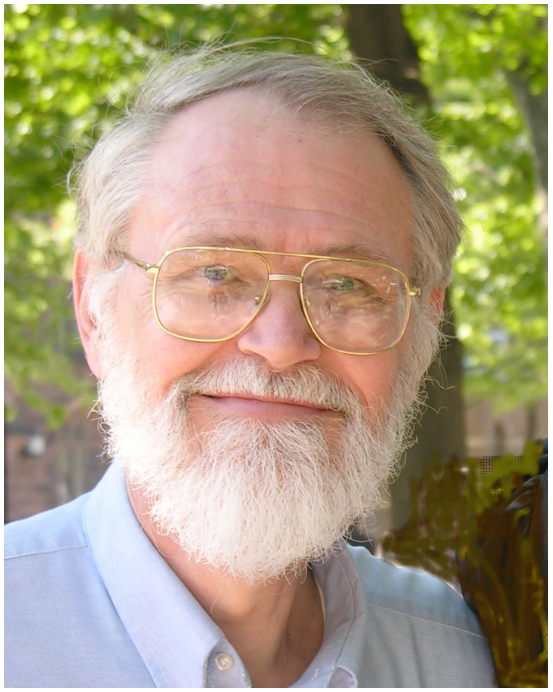
Course Outcomes

- Demonstrate a good understanding of C libraries for input and output, and the interface between C programs and the UNIX operating system.

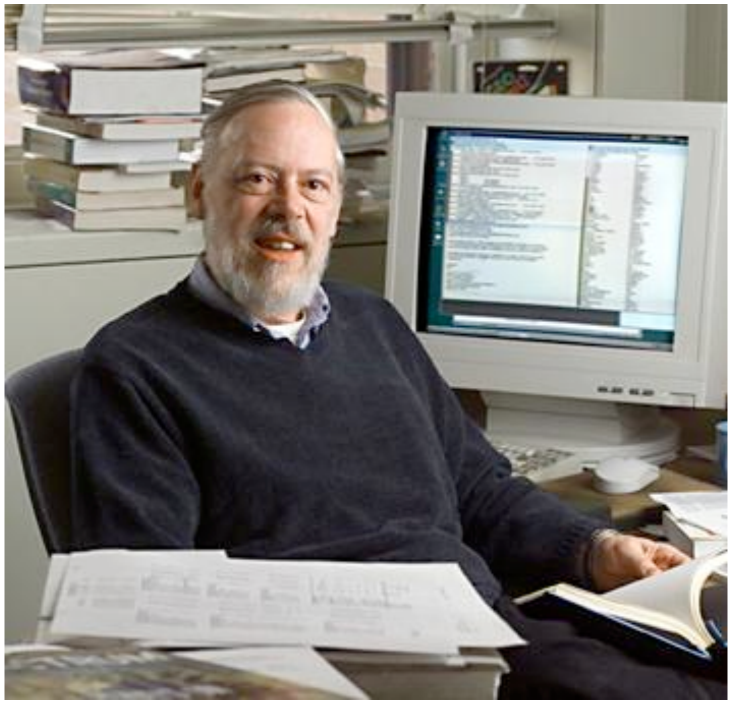
- Demonstrate an ability to use UNIX tools for program development and debugging
  - vi, emacs, jEdit

TextBook

- Brian Kernighan and Dennis Ritchie, The C Programming Language, 2nd ed., Prentice Hall, 1988 (a.k.a. K&R)



- Professor, Department of Computer Science Princeton University Princeton



- Dennis Ritchie from AT&T Bell Lab is the inventor of C

Topics

- C Types, Operators, and Expressions
- Control Flow
- Functions and Program Structures
- Pointers and Arrays
- Dynamic memory allocation
- Structures
- Bitwise operations
- Input and Output Libraries
- The Unix System Interface

Grading

labs: 20%

- lab attendance is mandatory

quizzes: 10%

- will be occasionally given during labs

projects: 30%

midterm exams: 20%

final exam: 20%

Policies

- All required assignments should be completed by the stated due date and time
- The total score of your assignment score will be 10 points less every extra day after the due date
  - i.e., the 100 total points will become zero after 10 days pass the due date
- You are responsible for keeping backups of your work
  - my disk crashed" and "my roommate ate my program" are not reasons for late submissions

Policies

- You can only turn in a program once.
- No revisions or additions can be made to your program after it has been submitted.

Policies

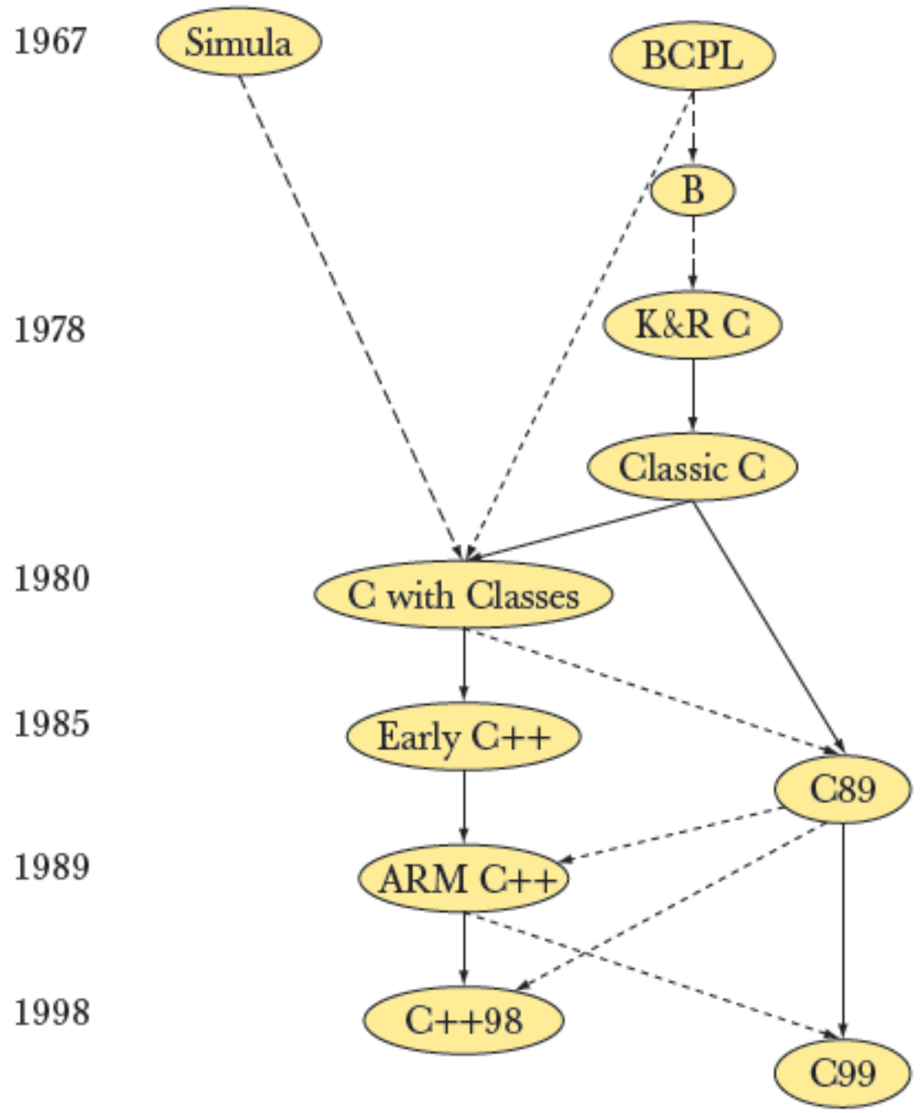
- All coursework is to be done independently
- You are encouraged to discuss the material BEFORE you do the assignment
- The homework should be written strictly by yourself
- Plagiarizing the homework will be penalized by maximum negative credit and cheating on the exam will earn you an F in the course.

A bit History

- born in the Computer Science Research Department of Bell Labs in Murray Hill, NJ

A bit of History

- C was created with Unix in mind



A bit of History

- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C
- International standard (ISO) in 1990 which was adopted by ANSI and is known as C89
- As part of the normal evolution process the standard was updated in 1995 (C95) and 1999 (C99)
- C++ and C
  - C++ extends C to include support for Object Oriented Programming and other features that facilitate large software development projects
  - Unfortunately, there are two ISO committees for C and C++.

Elements of a C Program

- A C development environment includes
  - **System libraries** and headers: a set of standard C libraries and their header files.
    - For example see /usr/include and glibc.
  - **Application Source**: application source and header files
  - **Compiler**: converts **source** to **object** code for a specific platform
  - **Linker**: resolves external references and **produces the executable** module

Elements of a C Program

- There must be one main function where execution begins when the program is run.
  - int main (void) { ... },
  - int main (int argc, char *argv[]) { ... }
  - UNIX Systems have a 3rd way to define main(), though it is not POSIX.1 compliant
    - int main (int argc, char *argv[], char *envp[])
- Preprocessors
  - macros, compiler controls, constant values
- additional local and external functions and variables

Example Code

- see example code

Pitfalls of C

- Great power comes with great responsibility
- C is procedural language, it easy to write spaghetti code
- Preprocessors can get really messy
- no way to gracefully terminate a program
  - no catch/throw/exception
- not too many (there are some) help from the language for doing OOP/OOD
- Many others... (Recommend reading: ***C Traps and Pitfalls***, by Andrew Koenig)

Your Tasks This Week

- We will use only **gcc**
- Your assignments will be compiled using gcc

- gcc is available on (virtually) all systems. This includes the
  - mason cluster,
  - Linux,
  - Windows (you must install Cygwin), and
  - Mac OS X (you must install Xcode).

Your Tasks This Week

- Learn basic Unix commands: ls, pwd, cp, cat, less, scp, ssh
- Learn about Makefile
  - GNU `make'
  - a power build system
  - determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them
  - this is very useful if you have many header files and source files