CS 4400
Computer Systems

LECTURE 1

What to expect from 4400

The life of a program

Administrative

Instructor:

Saday Sadayappan, saday@cs.utah.edu
Office Hours: Mon,Tue,Wed,Thu: 4:30-6pm

Teaching Assistants:

Sachin Boban, John Jolly, Calvin Lee, Joanna Lowry, Jaecee Naylor, Sirius Shahini, Guy Watson

Office Hours: TBA

All course information will be managed with Canvas: https://utah.instructure.com/courses/638167/

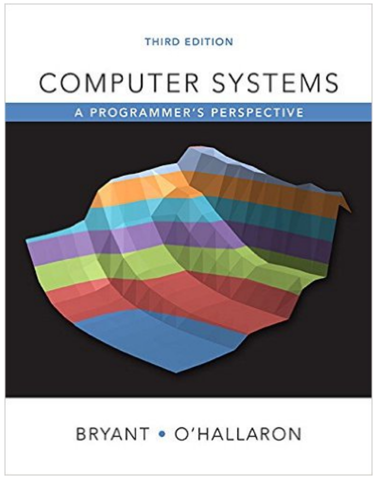
Acknowledgment

- Lecture materials from previous instructors of CS4400
  - Prof. Matthew Flatt
  - Prof. Erin Parker
  - Prof. Mary Hall

Background

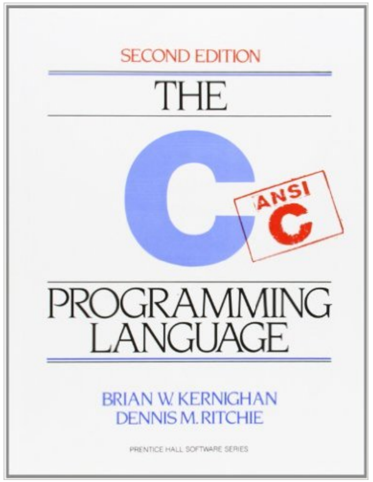
- The purpose of this course is to help you bridge the gap between high-level programming and the actual computer system.
  - “from a programmer’s point of view”
- A computer system consists of hardware and software working together to run application programs.
  - processors, memory, OS, compilers, networks, …
- Students must have passed CS 3810.
  - Students planning to take CS 3505 are <u>strongly recommended</u> to do so *before* CS 4400.
- Familiarity with C++ is assumed. This course uses C.

Textbook



Required:

Computer Systems, A Programmer's Perspective, Randal E. Bryant and David R. O'Hallaron, 3rd edition, Prentice-Hall, 2016 (ISBN: 0-13-409266-X).



Recommended:

The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie, 2nd edition, Prentice-Hall, 1988(ISBN: 978-0131103627).

What to expect from 4400

- Heavy use of C, “Unix”, and x86
- C vs. Java / C#
  - similar syntax and control statements
  - C: pointers, explicit dynamic memory allocation, formatted I/O, little support for abstractions (no classes)
- Despite what you have heard about CS 4400 …
  - Start all assignments as soon as they go live — there are only six and each is out 2-3 weeks.
  - Working at a low level can be frustrating — but unless you are working close to the deadline, we can help you get unstuck.
  - Our Industry Advisory Board and interviewers value skills learned in this course.

Life of a program — Source

- Example:

```
#include <stdio.h>

int main(void) {
  printf("hello, world\n”);

  return 0;
}
```

hello.c

- How is this program “born”?

```
2369 6e63 6c75 6465 203c 7374 6469 6f2e
683e 0a0a 696e 7420 6d61 696e 2876 6f69
6429 207b 0a20 2070 7269 6e74 6628 2268
656c 6c6f 2c20 776f 726c 645c 6e22 293b
0a0a 2020 7265 7475 726e 2030 3b0a 7d0a
```

hello.c in hexl-mode (emacs)

23 (i.e., 0x23) is the ASCII code for ‘#’ in hex

Life of a program — Translation

The compiler driver translates a source code file into an executable object file.

- preprocessing
- compilation — translates *hello.c* into *hello.s*
- assembly — translates *hello.s* into *hello.o*
- linking — links *hello.o* to the standard C library to get *hello*

```
unix> gcc -o hello hello.c
```

```
…
pushq   %rbp
movq    %rsp, %rbp
movl    $.LC0, %edi
call    puts
movl    $0, %eax
popq    %rbp
ret
```

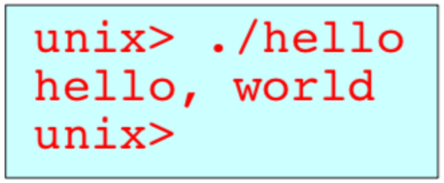
hello.s (x86 assembly language)

Compiler ≠ black box

- Is a *switch* statement always faster than *if else if* ?
- Should you use a recursive or an iterative solution?
- Are pointers more efficient than array indexes?
- Is it faster to access a local variable than an argument passed by reference?
- What does it mean when a symbol is undefined?
- What is the difference in static and dynamic libraries?
- How can buffer overflow corrupt the call stack?

Life of a program — Execution

- The shell is a command-line interpreter and assumes that the first word of a command line is a built-in shell command or the name of an executable file.



```
unix> ./hello
hello, world
unix>
```

Unix shell

- The shell loads the executable file *hello* in the current directory, copying the code and data from “disk” to main memory.

- The processor executes the instructions in main memory.

Operating system

- *hello* requires the keyboard, display, disk, and main memory — but none are accessed directly.
- The OS is a layer of software between the application program *hello* and the hardware.
- Three fundamental abstractions:
  - Process — provides the illusion that *hello* is the only program running on the system
  - Virtual memory — provides each process with the illusion that it has exclusive use of main memory
  - Files — every I/O device is viewed uniformly as a file

CS 4400 outcomes

- You will be a more effective programmer.
  - detecting and fixing bugs more efficiently
  - understanding and tuning program performance
  - being familiar with C and assembly code
- You will be more resourceful and self sufficient.
- You will be comfortable using the terminal / shell and command line.
- You will have a firm foundation for specialized systems classes and real-world software development.

Administrative

- Dr. or Prof. Saday or just Saday (he/him/his)
- Exam dates: Midterms on 9/30 and 11/2; Final 12/11
- Check Canvas and Piazza often — multiple times/ week
- Watch videos / read textbook + take quiz *before* every lecture (by 9am on day of lecture: hard deadline)
- Read and bring any questions about the Academic Misconduct Policy to the next lecture

Rest of Today’s Class

- Review syllabus, posted on Canvas
- Simple C programs