Computer Systems

COMP 2400

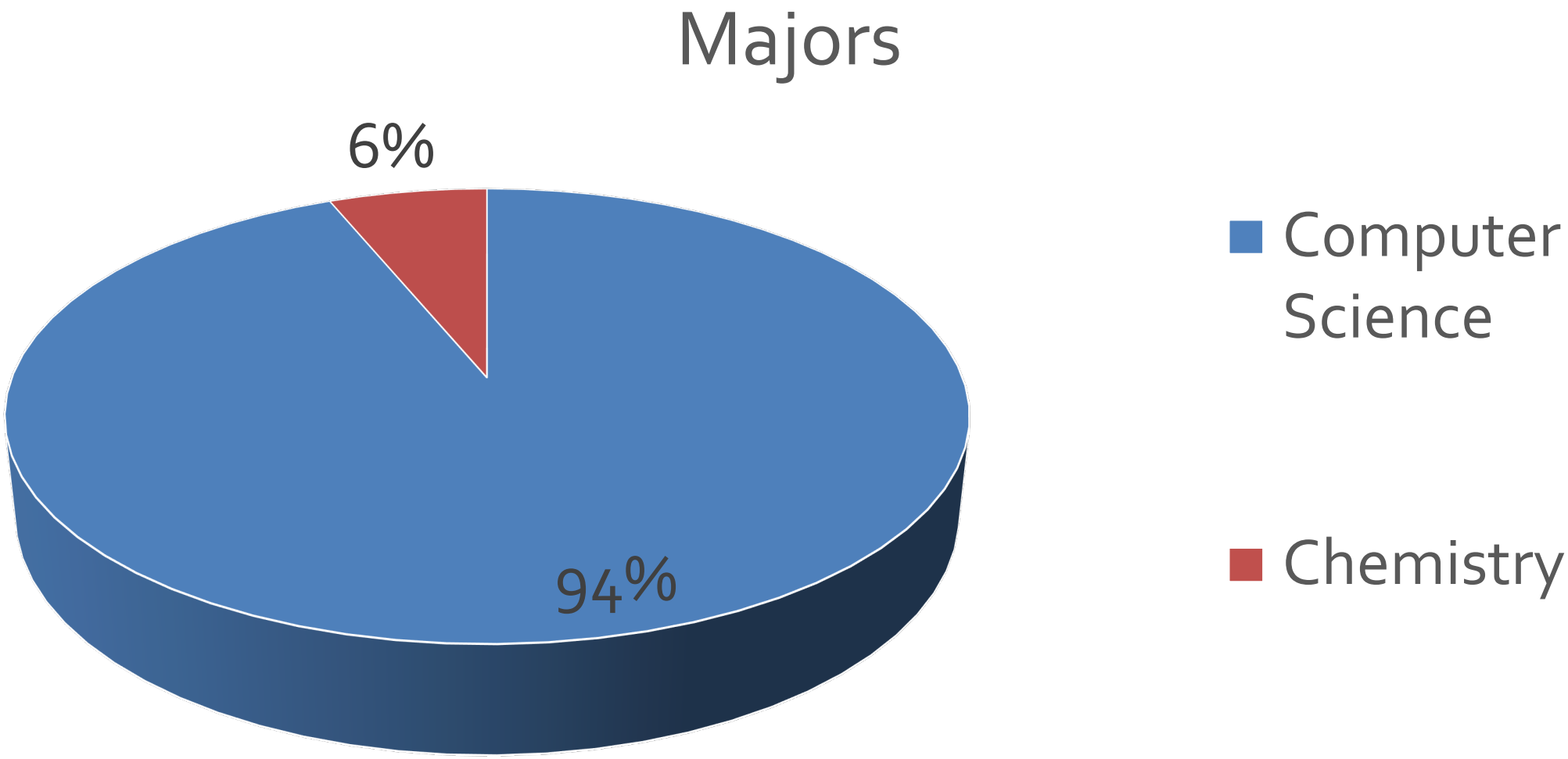
Who am I?

- Dr. Barry Wittman
- Not Dr. Barry Whitman
- Education:
  - PhD and MS in Computer Science, Purdue University
  - BS in Computer Science, Morehouse College
- Hobbies:
  - Reading, writing
  - Enjoying ethnic cuisine
  - DJing
  - Lockpicking
  - Stand-up comedy

How can you reach me?

- **E-mail:** wittman1@otterbein.edu
- **Office:** Art & Communication C123
- **Phone:** (614) 823-2944
- **Office hours:** **MWF** 10:15 – 11:15 a.m.,
  MW 3:00 – 4:00 p.m.,
  F 3:00 – 5:00 p.m.,
  T 10:00 – 11:15 a.m.,
  TR 2:00 – 4:00 p.m.,
  and by appointment
- **Website:**
  http://faculty.otterbein.edu/wittman1/

Who are you?



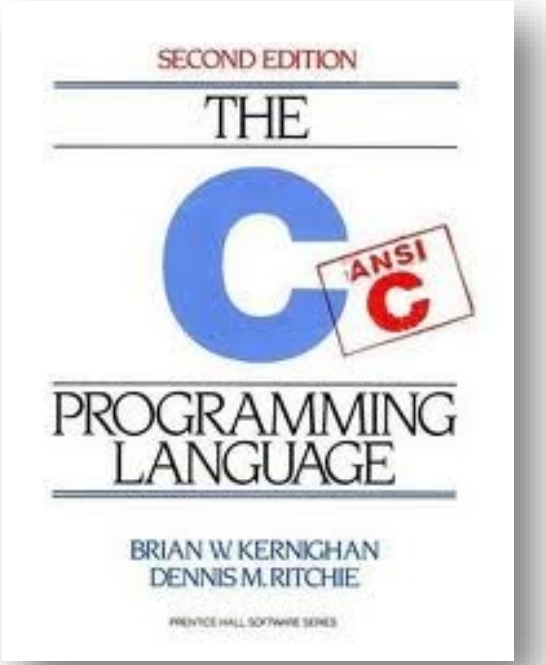
Why are we here?

- What's the purpose of this class?
- What do you want to get out of it?
- Do you want to be here?

Course Overview

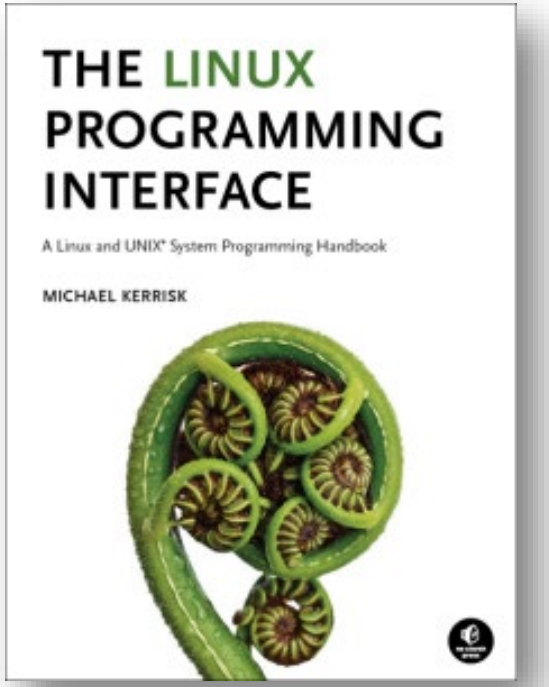
Textbooks

- Brian W. Kernighan and Dennis M. Ritchie
- ***The C Programming Language***
- 2nd Edition, 1988, Prentice Hall
- ISBN-10: 0131103628
- ISBN-13: 978-0131103627
- **Required textbook**
- The book that every serious computer scientist must have a copy of



Textbooks

- Michael Kerrisk
- ***The Linux Programming Interface***
- First Edition, 2010, No Starch Press
- ISBN-10: 1593272200
- ISBN-13: 978-1593272203
- Amazing book that you'll want to keep in your bag of tricks for all your future Linux hacking
- **Optional textbook**



You have to read the book

- You are expected to read the material before class
- If you're not prepared, you might be asked to leave
- You might forfeit the education you have paid around **$100 per class meeting** to get!

Course focuses

- C expertise
  - Another language in your tool belt
- Deeper knowledge of CPU and memory management
- Better understanding of the underlying OS
- Linux proficiency
- Command line tools
- Loving your inner geek

More information

For more information, visit the webpage:

`http://faculty.otterbein.edu/wittman1/comp2400`

- The webpage will contain:
  - The most current schedule
  - Notes available for download
  - Reminders about exams and homework
  - Syllabus
  - Detailed policies and guidelines

Projects

Six projects

- 36% of your grade will be six equally weighted projects
- Each will focus on a different major area from the course:
  - Basic math and I/O
  - Bitwise operations
  - String manipulation
  - Memory allocation
  - Dynamic data structures
  - Socket communication
- You will work on each project in two-person teams

Teams

- All projects are done in teams of two
- You may pick your partners
  - But you have to have a different partner for each project!
  - Use Brightspace to form teams
- Projects must be uploaded to Brightspace: https://otterbein.brightspace.com

Turning in projects

- Projects must be uploaded to Brightspace **before** the deadline
- Late projects will not be accepted
  - Exception: Each person will have 3 grace days
  - You can use these grace days together or separately as extensions for your projects
  - You must inform me **before** the deadline that you are going to use grace days
  - If two people in a team don't have the same number of grace days, the number of days they will have available will be the **maximum** of those remaining for either teammate

Labs

In-class Programming Exercises

Labs

- 15% of your grade will be based around programming labs
- Labs are on Tuesdays and Thursdays
- 15 of these labs will focus on the solution of a problem with a graded exercise
- Work should be done individually, but the goal is to learn, and I will help everyone
- The remaining lab days are to discuss course material and work on team projects
- **You are expected to attend all lab days**

Tickets Out the Door

Tickets out the door

- 5% of your grade will be tickets out the door
- These tickets will be based on material covered in the previous one or two lectures
- They will be graded leniently
- They are useful for these reasons:
  1. Informing me of your understanding
  2. Feedback to you about your understanding
  3. Easy points for you
  4. Attendance

Exams

Exams

- There will be two equally weighted in-class exams totaling 30% of your final grade
  - **Exam 1:** 02/17/2025
  - **Exam 2:** 03/24/2025
- The final exam will be worth another 14% of your grade
  - **Final:** 8:00 – 10:00 a.m.

    5/01/2025

Course Schedule

Tentative schedule

Week	Starting	Topics	K & R Chapters	LPI Chapters	Notes
1	01/13/25	Introduction	1	1	
2	01/20/25	Data representation	2	11	MLK, Project 1 Due
3	01/27/25	Control flow	2, 3	2, 3	
4	02/03/25	Functions	4	6	Project 2 Due
5	02/10/25	Arrays and Strings	4, 5		
6	02/17/25	Pointers	5		Exam 1
7	02/24/25	Memory allocation	5	7	Project 3 Due
8	03/03/25	Structs	6	8, 10	
	03/10/25	**Spring Break**			
9	03/17/25	Advanced structs	6		Project 4 Due
10	03/24/25	Files and streams	7	4	Exam 2
11	03/31/25	File systems		5, 13, 14, 15	
12	04/07/25	Networking	5	56, 57, 58, 59	Project 5 Due
13	04/14/25	C++		Notes	Good Friday
14	04/21/25	Review	All	All	Project 6 Due

Project schedule

- **Project 1:** **6%** Tentatively due **01/24/2025**
- **Project 2:** **6%** Tentatively due **02/07/2025**
- **Project 3:** **6%** Tentatively due **02/28/2025**
- **Project 4:** **6%** Tentatively due **03/21/2025**
- **Project 5:** **6%** Tentatively due **04/09/2025**
- **Project 6:** **6%** Tentatively due **04/25/2025**

Policies

Grading breakdown

36%	• Six projects • Equally weighted
15%	• In-class labs
5%	• Tickets out the door
30%	• Two equally weighted midterm exams
14%	• Final exam

Grading scale

A	93-100	B-	80-82	D+	67-69
A-	90-92	C+	77-79	D	60-66
B+	87-89	C	73-76	F	60-62
B	83-86	C-	70-72		

Attendance

- You are expected to attend all classes and labs
- You are expected to have read the material we are going to cover **before** class
- Missed tickets out the door cannot be made up
- Exams and labs must be made up **before** the scheduled time, for excused absences

R-E-S-P-E-C-T

- I hate having a slide like this
- I ask for respect for your classmates and for me
- You are smart enough to figure out what that means
- A few specific points:
  - Silence communication devices
  - Don't play with your phones
  - Don't use the computers in class unless specifically told to
  - No food or drink in the lab

Computer usage

- We will be doing a lot of work on the computers together
- However, students are always tempted to surf the Internet, etc.
- Research shows that it is nearly impossible to do two things at the same time (e.g. watch TikTok and listen to a lecture)
- For your own good, I will enforce this by taking 1% of your final grade every time I catch you playing on your phones or using your computer for anything other than course exercises

Academic dishonesty

- Don't cheat
- **First offense:**
  - I will try to give you a zero for the assignment, then lower your final letter grade for the course by one full grade
- **Second offense:**
  - I will try to fail you for the course and try to kick you out of Otterbein
- Refer to the syllabus for the school's policy
- Ask me if you have questions or concerns
- **You are not allowed to look at another student's code, except for group members in group projects (and after the project is turned in)**
- **You may not use generative AI tools like ChatGPT to write *any* code you turn in for this class**
- **I will use tools that automatically test code for similarity**

AI statement

- Artificial Intelligence (AI) is any computer system designed to perform a cognitive or behavioral task historically believed to be one only humans can perform. Generative AI is a term used for recent AI systems that generate significant quantities of content such as text, images, audio, or video from a short input prompt, usually text.
- Although generative AI tools are impressive, they must not be used to write any code that a student is expected to turn in for this class. Generative AI tools may be used to explain existing code or to suggest improvements for code but only *after* the project or lab in question has been turned in. Students who do not write code themselves have missed the opportunity to gain the skills of logical problem solving and translation to a formal programming language that are essential for computer scientists. Submitting work that includes or is derived from AI-generated materials shall be considered an act of academic dishonesty.

Disability Services

- The University has a continuing commitment to providing access and reasonable accommodations for students with disabilities, including mental health diagnoses and chronic or temporary medical conditions
- Students who may need accommodations or would like referrals to explore a potential diagnosis are urged to contact Disability Services (DS) as soon as possible
- DS will facilitate accommodations and assist the instructor in minimizing barriers to provide an accessible educational experience
- Please contact DS at DisabilityServices@otterbein.edu
- More info can also be found at http://www.otterbein.edu/ods
- Your instructor is happy to discuss accommodations privately with you as well

C Basics

Types in C

- Basic types in C are similar to those in Java, but there are fewer

Type	Meaning	Size
`char`	Smallest addressable chunk of memory	Usually 1 byte
`short`	Short signed integer type	At least 2 bytes
`int`	Signed integer type	At least 2 bytes, usually 4 bytes
`long`	Long signed integer type	At least 4 bytes
`float`	Single precision floating point type	Usually 4 bytes
`double`	Double precision floating point type	Usually 8 bytes

- No built-in Boolean type in C89 (but more about that later)

But, wait, it gets worse …

- Unlike Java, C has signed and unsigned versions of all of its integer types
  - Perhaps even worse, there's more than one way to specify their names

Type	Equivalent Types
`char`	`signed char`
`unsigned char`	

`short`	`signed short`
	`short int`
	`signed short int`

`unsigned short`	`unsigned short int`
`int`	`signed int`
`unsigned int`	`unsigned`

`long`	`signed long`
	`long int`
	`signed long int`

`unsigned long`	`unsigned long int`

And yet again worse than that …

- There are also types that are officially supported in C99 but may or may not be supported by compilers in C89

Type	Meaning	Size
`long long`	Very long signed integer type	At least 8 bytes
`long double`	Extended precision floating point type	Usually 10 bytes or 16 bytes

- Naturally, a **`long long`** can also be written as a **`long long int`**, a **`signed long long int`** and has siblings **`unsigned long long`** and **`unsigned long long int`**

Bringing in `bool`

- There's a way to define types in C that we'll get into later
- Someone used it to create a **`bool`** type
- It's not part of the C language, but it's commonly used
- To use it, put **`#include <stdbool.h>`** at the top of your program
- Once you do that, you'll have access to the **`bool`** type and the constants **`true`** and **`false`**
- Note that **`bool`** is a synonym for **`_Bool`**, a special unsigned integer type added in C99
- The value of **`true`** is **`1`**, and the value of **`false`** is **`0`**

```
bool value = true;
bool compare = 4 < 2;
```

Derived types

- Structs
  - Collections of a fixed set of named items
  - Similar to a class with no methods and all public members
- Unions
  - A set of possible items, but only one of them is stored at a time
  - Used to conserve memory (but hard to program with)
- Arrays
  - Lists of items of with the same type
  - Can be indexed with integers
- Pointers
  - Types that point at other variables
  - Contain addresses
  - Pointer arithmetic is allowed, meaning that you can point at a variable, and then see what value exists 38 bytes later in memory

Hello, World

- The standard Hello World program is simpler in C, since no surrounding class is needed

```c
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

Includes

- Libraries written by other people (and eventually code you've written yourself) can be used in your program using the **`#include`** directive
  - Only include header files (**`.h`** extension)
  - **`stdio.h`** is the header for basic input and output methods
- Standard libraries are specified in angle brackets: **`<stdio.h>`**
- Local files are specified in quotes: **`"mycode.h"`**
- It's legal to put **`#include`** directives anywhere in the code, but it's good style to put them at the top

`main()` function

- Executable code in C is inside of **functions**
  - Functions are similar to methods in Java
  - Think of them as static methods, since none of them are in an object
- Execution starts at the **`main()`** function
- Traditionally, the **`main()`** function has the **`int`** return type and returns **`0`** at the end
  - A value of **`0`** tells the OS that the program exited without error
  - Some people prefer a **`main()`** with **`void`** as its return type

printf() function

- The **printf()** function is the classic console output function in C
- It always prints out a string
- The string can have special control characters inside of it that are used to print numbers or other strings, all with specified formatting
- Any number of arguments can be given after the initial string, provided that there is a format specifier for each one

```
printf("%d fish, %f fish", 1, 2.0);
printf("%s in socks", "fox");
```

Format specifiers

- These specifiers can be used in a **`printf()`** format string
- They are preceded by a percent sign (`%`)
- You can also specify a minimum width (after the `%`) and a specific precision (after a `.` and before the specifier)

Specifier	Output
`d, i`	Integer
`u`	Unsigned integer
`f`	Floating point number
`e`	Floating-point number with exponent
`g`	Floating-point number in standard or scientific notation depending on size
`x`	Unsigned integer in hexadecimal
`o`	Unsigned integer in octal
`s`	Null-terminated string
`c`	Character

```
printf("You owe me $%.2f in cash!", 50.0/3);
```

Text editors

- You're probably used to using IntelliJ for editing code
- In the Linux world, compilers are often separate from editors
- You can pick whichever text editor you like
- Ubuntu provides GNOME Text Editor
- **`vim`** and **`emacs`** are two editors that run from the command line and do not require a GUI
  - They take some getting used to but are very powerful

Navigating with the command line

- Click on the white dots in the lower left and type in "terminal" or just type Ctrl-Alt-t
- A command line will open up
- Type **`ls`** to list the current directory contents
- Type **`cd`** to change to another directory
  - **`cd ..`** changes to the parent directory

```
> cd stuff
> |
```

Compiling

- Navigate to whichever directory you saved your **`.c`** file
- Type **`gcc`** followed by the name of the file

```
> gcc hello.c
```

- By default, the executable will be called **`a.out`**
- To run your code, type **`./a.out`**
  - The **`./`** specifies the current directory

```
> ./a.out
```

Naming the output

- Typically, you'll name your output file rather than using the default **`a.out`**
- For example, to name your output **`hello`**, you put **`-o hello`** into the compilation command
  - The **`-o`** is for *output*

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

- To run **`hello`**, type **`./hello`**

```
> ./hello
```

Makefiles

- The order of compilation matters
- You have to compile all necessary files yourself to make your program work
- To make these issues easier to deal with, the **`make`** utility is used
- This utility uses makefiles
  - Each makefile has a list of targets
  - Each target is followed by a colon and a list of dependencies
  - After the list of dependencies, on a new line, preceded by a **tab**, is the command needed to create the target from the dependencies

Sample makefile

- Makefiles are called **`makefile`** or **`Makefile`**

```
all:    hello

hello: hello.c
    gcc -o hello hello.c

clean:
    rm -f *.o hello
```

Credits

- Much of the structure and content of this lecture is based on notes from Dennis Brylow from his version of CS240 taught at Purdue University

Upcoming

Next time…

- More C basics
- C compilation model
- History of Unix and Linux
- Please read K&R Chapter 1 and LPI Chapter 1
- Lab meets tomorrow
  - Come to get familiar with Linux
  - Choose your teammate for Project 1

Reminders

- Read K&R Chapter 1 and LPI Chapter 1
- Form your teams for Project 1
- Consider dual-booting Linux on your machine if you don't have it already
  - Another option is running Linux inside of Virtual Box