Operating Systems and Network Fundamentals

# COMP 3400

## 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

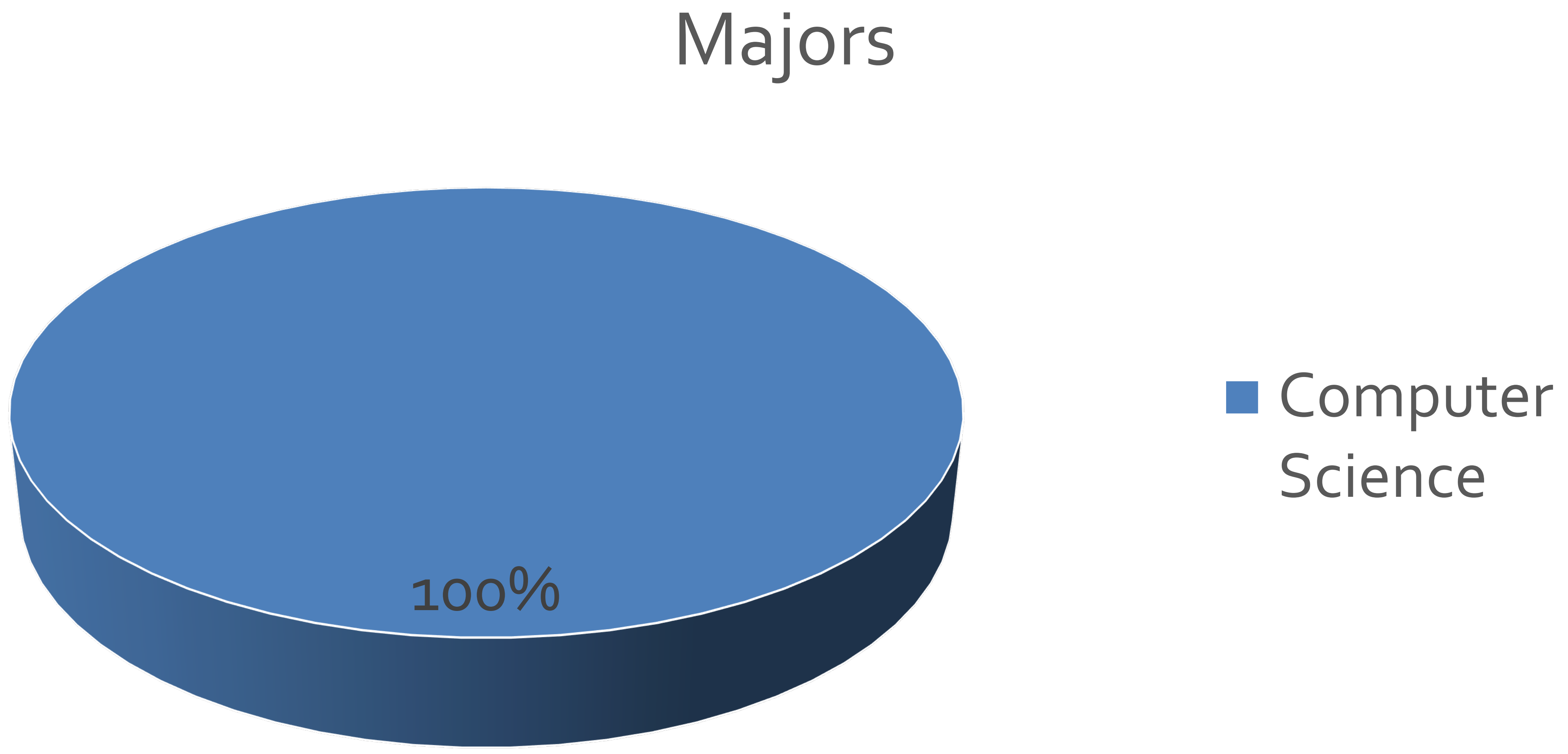
## How can you reach me?

- **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

## Textbook

- Michael S. Kirkpatrick
- ***OpenCSF: Computer Systems Fundamentals***
- Available: https://w3.cs.jmu.edu/kirkpams/OpenCSF/Books/csf/html/
- The book is free and includes interactive questions to test your knowledge at the ends of sections

## 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

- Deeper C expertise
- Linux system calls
- Processes
- Signals
- Interprocess communication
- Shared memory
- Threading
- Synchronization
- Network programming

## More information

For more information, visit the webpage:

`http://faculty.otterbein.edu/wittman1/comp3400`

- The webpage will contain:
  - The most current schedule
  - Notes available for download
  - Reminders about exams and homework
  - Syllabus (you can request a printed copy if you like)
  - Detailed policies and guidelines

## Projects

## Three projects

- 27% of your grade will be three equally weighted projects
- Each will focus on a different topic:
  - Function pointers and finite state machines
  - Processes and intrusion detection
  - Networking and CGI
- 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

## 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

## Project framework

- Projects (and assignments) work differently than in my other classes
- Relatively large frameworks of skeleton code will be given to you
- Unit tests and integration tests will also be provided
- Understanding the tests will help you understand what you need to code
- You won't have to write much code … but it will have to be code that you understand well

# Running tests

- Inside the top-level project directory, type **`make test`** to run the tests
- The top-level directory will contain a **`tests`** directory with lots of important stuff:
  - **`tests/public.c`**
    - Driver for unit tests, using the Check framework for unit testing
    - Some tests will be given, but you should add more
  - **`tests/itests.include`**
    - Configuration file that gives command-line arguments for integration testing.
    - You can modify this file to add test cases for both good and bad command-line arguments.
  - **`tests/expected/`**
    - Directory contains text files with the expected output for integration tests.
    - When you add test cases to **`itests.include`**, you must also create a corresponding **`.txt`** file in this directory
    - **`diff`** is used to check, so output must match to the character
  - **`tests/inputs/`**
    - Directory contains files that can be used as input to the projects
  - **`tests/Makefile`**, **`tests/integration.sh`**, and **`tests/testsuite.c`**
    - Drivers for the testing infrastructure that you don't need to modify

## Assignments

## Assignments

- 24% of your grade will be single-week programming assignments
- These assignments are grouped:
  - Assignments 1 and 2 are grouped with Project 1
  - Assignments 3 and 4 are grouped with Project 2
  - Assignments 5 and 6 are grouped with Project 3
  - Assignments 7 and 8 are grouped together without a project
- Because the code is interrelated, you will have the same teams for each grouping of assignments and projects
- Assignments are intended to make the projects easier
  - **Do the grouped assignments first before starting on the project!**

## Turning in assignments

- Assignments must be uploaded to Brightspace **before** the deadline
- Late assignments will not be accepted
- There are no grace days for assignments

## 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.

    4/30/2025

## Course Schedule

# Tentative schedule

| Week | Starting | Topics | OpenCSF Chapters | Notes |
|---|---|---|---|---|
| **1** | 01/13/25 | Introduction | 1 | |
| **2** | 01/20/25 | Kernel and System Calls | 2 | Assignment 1 |
| **3** | 01/27/25 | Processes, files, and signals | 2 | Assignment 2 |
| **4** | 02/03/25 | IPC | 3 | Project 1 |
| **5** | 02/10/25 | Shared Memory | 3 | Assignment 3 |
| **6** | 02/17/25 | Networking | 4 | Assignment 4 |
| **7** | 02/24/25 | More Networking | 4 | |
| **8** | 03/03/25 | Internet | 5 | Project 2 |
| | 03/10/25 | **Spring Break** | | |
| **9** | 03/17/25 | Threading | 6 | Assignment 5 |
| **10** | 03/24/25 | Synchronization Primitives | 7 | Assignment 6 |
| **11** | 03/31/25 | More on Synchronization Primitives | 7 | |
| **12** | 04/07/25 | Synchronization Problems | 8 | Project 3 |
| **13** | 04/14/25 | Parallel and Distributed Systems | 9 | Assignment 7 |
| **14** | 04/21/25 | Review | All | Assignment 8 |

## Project schedule

| | | |
|---|---|---|
| **Project 1:** | **9%** | Tentatively due **02/07/2025** |
| **Project 2:** | **9%** | Tentatively due **03/07/2025** |
| **Project 3:** | **9%** | Tentatively due **04/11/2025** |

## Assignment schedule

|  |  |  |
|---|---|---|
| **Assignment 1:** | **3%** | Tentatively due **01/24/2025** |
| **Assignment 2:** | **3%** | Tentatively due **01/31/2025** |
| **Assignment 3:** | **3%** | Tentatively due **02/14/2025** |
| **Assignment 4:** | **3%** | Tentatively due **02/21/2025** |
| **Assignment 5:** | **3%** | Tentatively due **03/21/2025** |
| **Assignment 6:** | **3%** | Tentatively due **03/28/2025** |
| **Assignment 7:** | **3%** | Tentatively due **04/17/2025** |
| **Assignment 8:** | **3%** | Tentatively due **04/25/2025** |

## Policies

## Grading breakdown

| | |
|---|---|
| 27% | • Three projects<br>• Equally weighted |
| 24% | • Eight assignments<br>• Equally weighted |
| 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
- 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 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 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)**
- **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 assignment 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 here. Your instructor is happy to discuss accommodations privately with you as well.

## GNU Style

# GNU style

- In most classes, I let you make a lot of choices about style
  - For example, should you have braces on the same line as the header or the next line?
  - I emphasize **consistency**
- In this class, however, you have to use GNU style for C
- It's different from any style you've probably used before
  - And ugly!
- But it has value for several reasons:

1. It's a real standard used for GNU projects, including a huge number of open-source projects
2. You might be forced by your employer to adhere to some arbitrary standard in the future
3. Being able to adopt a particular standard as needed is a good skill for a professional software engineer

## Spaces vs. tabs

- There's a long-standing, quasi-religious debate over whether code should be indented with spaces or tabs
  - https://www.youtube.com/watch?v=oRva7UxGQDw
- Because of the Java IDEs we use, you're probably used to tabs
- **Bad news:** GNU style uses exactly two spaces for all indentation
- **Good news:** Your development practices don't need to change much.
- Tab (`\t`) is a character, but it doesn't have to be the character that pops out when you press the tab key on your keyboard
- Most development environments allow you to specify that hitting the tab key can produce a tab or a specific number of spaces
- You can configure **`gedit`** (or whatever IDE you're using) to output 2 spaces whenever you hit tab

## Braces and indentation

- In GNU style, pairs of braces should be in the same column, one below the other, unlike the Java standard of putting opening braces on the same line as the header
- Blocks like functions, selection statements, and loops should be indented with 2 spaces
  - If you're indenting a single line, that's all you need to do
  - However, selection statements and loops have their braces indented and the contents of those braces indented *again*

```
if (i < 10)
  {
    printf ("%d\n", i);
  }
else
  {
    printf ("Too big\n");
  }
```

## More on braces and indentation

- Another peculiarity of GNU style is that the return types of functions are written on the line *before* the function name, making the function name the first thing on a line

```
int
main (int argc, char **argv)
{
   /* Code here */
   return 0;
}
```

## Tools to help

- First, the test suite included with every project and assignment will check for compliance with GNU style and complain about every file that doesn't match
- Second, there's a magical tool called **`clang-format`** that can actually *convert* your code into GNU style (or a bunch of other styles)
- Example using **`clang-format`** to convert something called **`program.c`** to GNU style:

```
> clang-format --style=gnu -i program.c
```

# More information on style

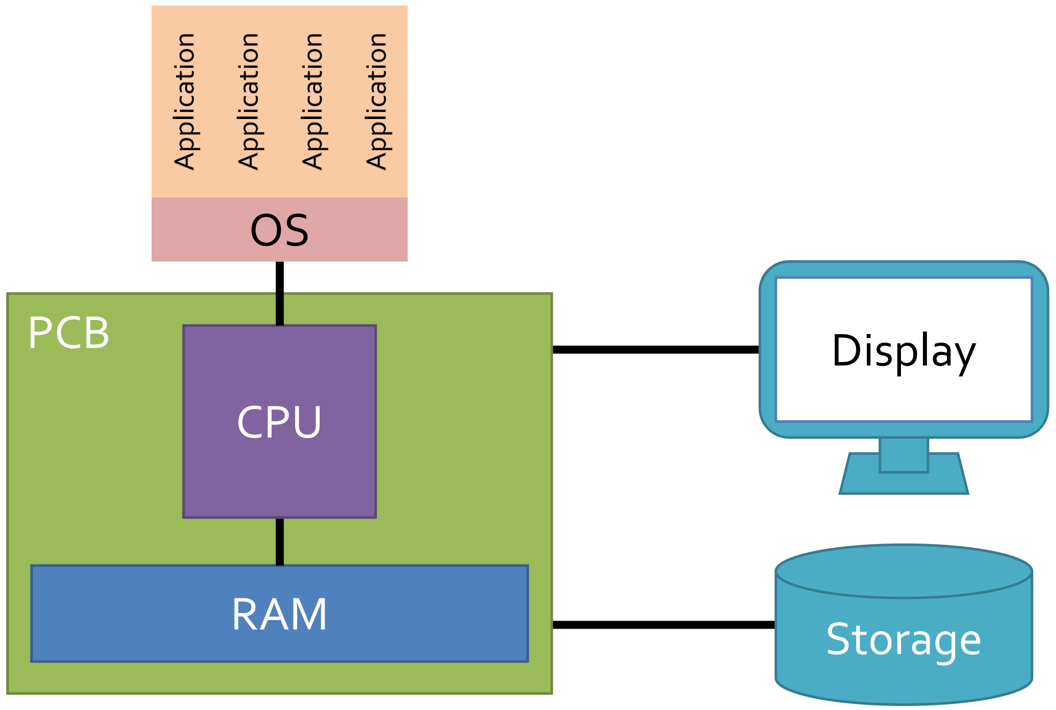
- For a more explanation and examples of the style you're expected to use, please visit the COMP 3400 Standards page:
  - http://faculty.otterbein.edu/wittman1/comp3400/standards/
- On a related note, even the best CS programs don't always have time to help students learn all the tools they'll need to use: editors, scripting, version control, debugging, etc.
- At MIT, some people put together a series of video and text lectures explaining what they think are some of the most important tools:
  - https://missing.csail.mit.edu/

## Systems and Models

## Computer systems

- The word "systems" comes up in computer science all the time
  - It means nothing, and it means everything
- We can think of a computer system as a collection of interacting components

- Hardware:
  - CPU
  - Storage
  - I/O
  - RAM
  - All attached via a PCB
- Software:
  - OS talking to the hardware
  - Applications



## Systems of systems

- Just as one computer can be thought of as a system, we can also network computers together to form systems of systems
- There are problems communicating within a single computer
- These problems happen at a larger scale when communicating and coordinating between many computers
- Traditional OS course focus almost exclusively on process scheduling and memory management within a single computer
- This course focuses on the same fundamental problems but at several levels of implementation
- Its goal is to make you a better programmer rather than an expert on OS internals

## Models

- Systems are complex
- Thus, we make **models**, simplified representations of systems
- These models are often visual, taking the form of labeled boxes with arrows
  - But formal models like equations or statements of logic are common too
  - If you've taken COMP 3100, you're familiar with UML, a standard visual way to model systems in CS
- The **level of abstraction** means how much detail has been removed
  - A high level of abstraction means we're focusing on the essentials of the system
- As computer scientists, we often have to turn models into code

## Upcoming

## Next time…

- Course themes
- System architectures

## Reminders

- Read sections 1.3 and 1.4
- If you're rusty on C, read Appendix A
- Look over Assignment 1
  - Due **next** Friday
- Form teams for Assignments 1 and 2 and 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