CMSC201
Computer Science I for Majors

Introduction

Prof. Jeremy Dixon

Introductions

- Professor Jeremy Dixon
  - Education
    - DSc in Information Technology (Towson) - ABD
    - MS in Information Technology (Hopkins)
    - MBA (Hopkins)
    - MS in Geoenvironmental Studies (Ship)
  - Likes:
    - Long Walks on the Beach
    - Running
    - Video Games

Course Overview

Course Information

- First course in the CMSC intro sequence
  - Followed by 202
- CS majors must pass with a B or better
- CMPE majors must get at least a C
- No prior programming experience needed
  - Some may have it

What the Course is About

- Introduction to Computer Science
  - Problem solving and computer programming
- We're going to come up with algorithmic solutions to problems
  - What is an algorithm?
- We will communicate our algorithms to computers using the Python language

Class Objectives

- By the end of this class, you will be able to:
  - Use an algorithmic approach to solve computational problems
  - Break down complex problems into simpler ones
  - Write and debug programs in the Python programming language
  - Be comfortable with the UNIX environment

Why Learn to Program?

- Programming skills are useful across a wide range of fields and applications
  - Many scientific professions utilize programming
  - Programming skills allow you to understand and exploit “big data”
  - Logical thinking learned from programming transfers to many other domains

Grading Scheme

- This class has:
  - 8 Homeworks (4% each)
    - small programming assignments
  - 2 Projects (8% each)
    - larger programming assignments
  - 10 lab/discussion sections (1% each)
  - 2 mandatory surveys (1% each)
  - A midterm (15%)
  - A comprehensive final exam (25%)

A Note on Labs

- Your “discussion” section is actually a lab
  - In the Engineer building (021, 104, 104A, 122)
- Labs are worth 10% of your grade
- You must attend your **assigned** section
  - No points for attending other sections

Submission and Late Policy

- Homeworks and projects will be submitted over the GL server with the submit command

- Homeworks will always be due at 9 pm
- Late homeworks will receive a ***zero***
- (In other words, there are no late homeworks)

Submission and Late Policy

- It is not recommended that you submit close to the deadline
  - Sometimes the server gets overloaded with everyone trying to submit
- Developing programs can be tricky and unpredictable
  - Start early and submit early (and often)

Academic Integrity

Academic Integrity

- We have homeworks and projects in this class
- You should never, *ever,* ***ever*** submit work done by someone else as your own.
- If you submit someone else's code, both students will get a 0 on the assignment.
  - Reminder: this a B-to-progress class for CMSC majors!

Things to Avoid

- Copying and pasting another student's code
- Leaving your computer logged in where another student can access it
- Giving your code to another student
- Attempting to buy code online
  - This will result in an immediate F in the class

Things that are Okay

- And encouraged!

- Talking to your friends about a problem
- Helping a fellow student debug (as long as your hands don't touch the keyboard!)
- Getting help from a TA or tutor

Why So Much About Cheating?

- Every semester, around 20 students get caught sharing code. Typically, they are stressed, confused, and just wanted to take a shortcut or help a friend. These students endanger their entire academic career when they get caught.
- If you feel like you can't possibly finish a project or homework on your own, contact someone in the course staff for help.

Getting Help

Where to Go for Help

- There are a number of places you can go if you are struggling!
  - All of our TAs happy to help.
  - If the TAs aren't working out, come by the professors' office hours (this should not be your first resort for help)
- All office hours are posted on the website.

Additional Help

- Tutoring from the Learning Resources Center
  - By appointment
- Computer help from OIT
  - By phone or in person
- See the syllabus on Blackboard for more info

Announcement: Note Taker Needed

A peer note taker has been requested for this class. A peer note taker is a volunteer student who provides a copy of his or her notes for each class session to another member of the class who has been deemed eligible for this service based on a disability. Peer note takers will be paid a $200 stipend for their service. Peer note taking is not a part time job but rather a volunteer service for which enrolled students can earn a stipend for sharing the notes they are already taking for themselves.

If you are interested in serving in this important role, please fill out a note taker application on the Student Support Services website or in person in the SSS office in Math/Psychology 213.

UMBC Computing Environment

- We develop our programs on UMBC's GL system
  - GL is running the Linux Operating System
    - GUI – Graphical User Interface
    - CLI – Command-Line Interface
- Lab 1 will walk you through using the UMBC computing environment

How Do I Connect to GL?

- Windows
  - Download Putty (Lab 1 has a video about this)
  - Hostname – gl.umbc.edu
  - Make sure you pick SSH
  - Put in username and password
- Mac
  - SSH client already installed
  - Go to the Application folder and select Utilities
  - Open up a terminal window
  - Enter the following: **`ssh -l <username> gl.umbc.edu`**
  - Put in your password

Linux Commands

- See: http://www.csee.umbc.edu/resources/computer-science-help-center/#Resources

- Here's a few basic commands:

  `ls` – list contents

  – List files and directories in your current directory

  – Directory is just another word for folder

More Basic Commands

- **Important!!** Commands are case sensitive

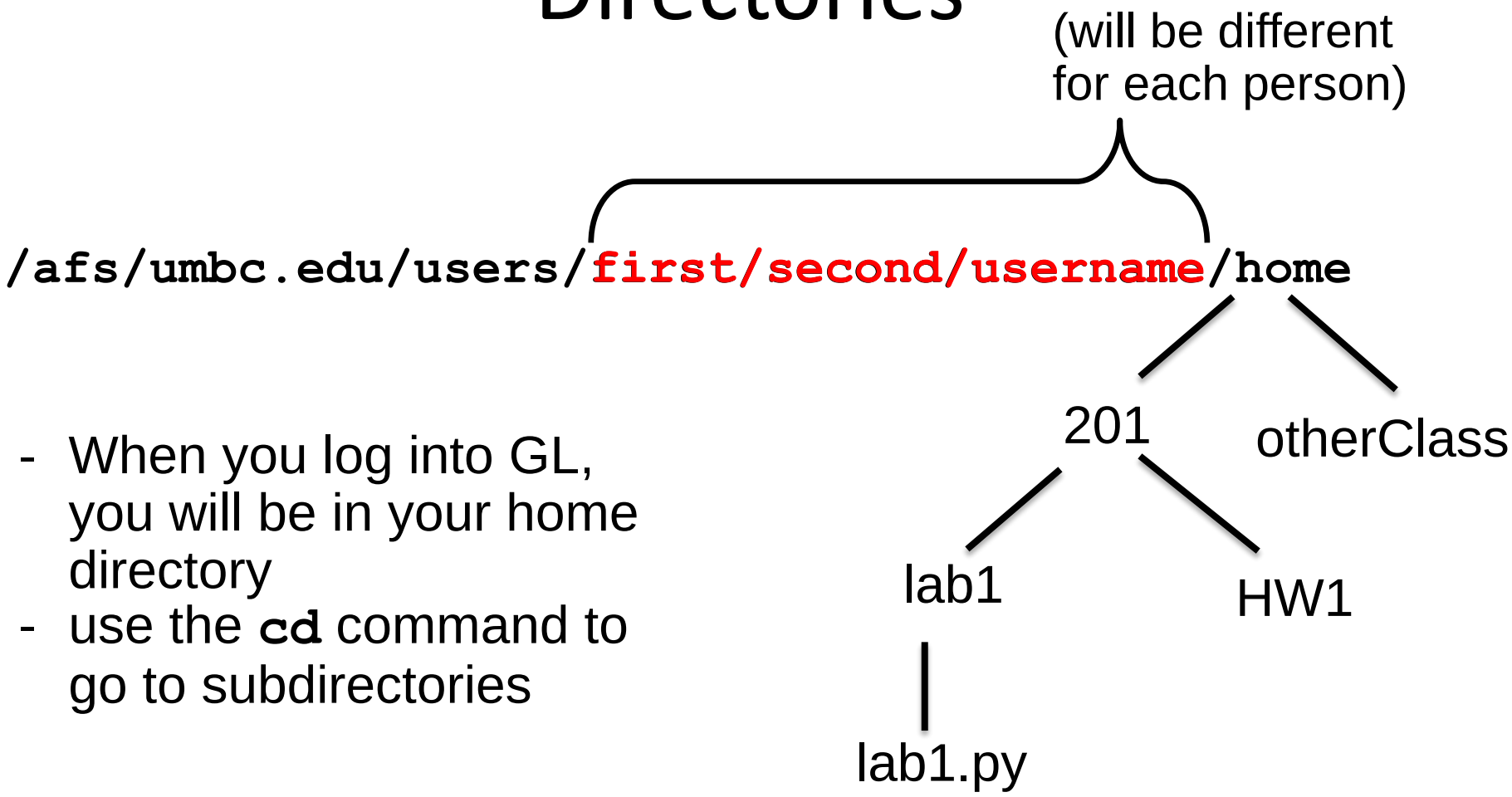
`cd <name>` – change directory

`cd ..` – go to parent directory

`cd .` – stay in current directory

`mkdir <name>` – make a new directory

Directories



- When you log into GL, you will be in your home directory
- use the **`cd`** command to go to subdirectories

emacs – A Text Editor

- Will use emacs to write our python code
- emacs is CLI, not GUI
  - Need to use keyboard shortcuts to do things
- Reference:
  - http://www.csee.umbc.edu/summary-of-basic-emacs-commands/

Keyboard Shortcuts for emacs

- To open a file (new or old)

  `emacs filename_goes_here.txt`

- To save a file

  `CTRL+X` then **`CTRL+S`**

- To save and close a file

  `CTRL+X` then **`CTRL+C`**

- To undo

  `CTRL+_` (that “CTRL + Shift + -” for underscore)

Computers and Programs (Zelle Chapter 1)

Today's Objectives

- To have a very basic overview of the components of a computer system
- To understand how data is represented and stored in memory
- To be aware of elements of the UMBC computing environment
- To start thinking algorithmically

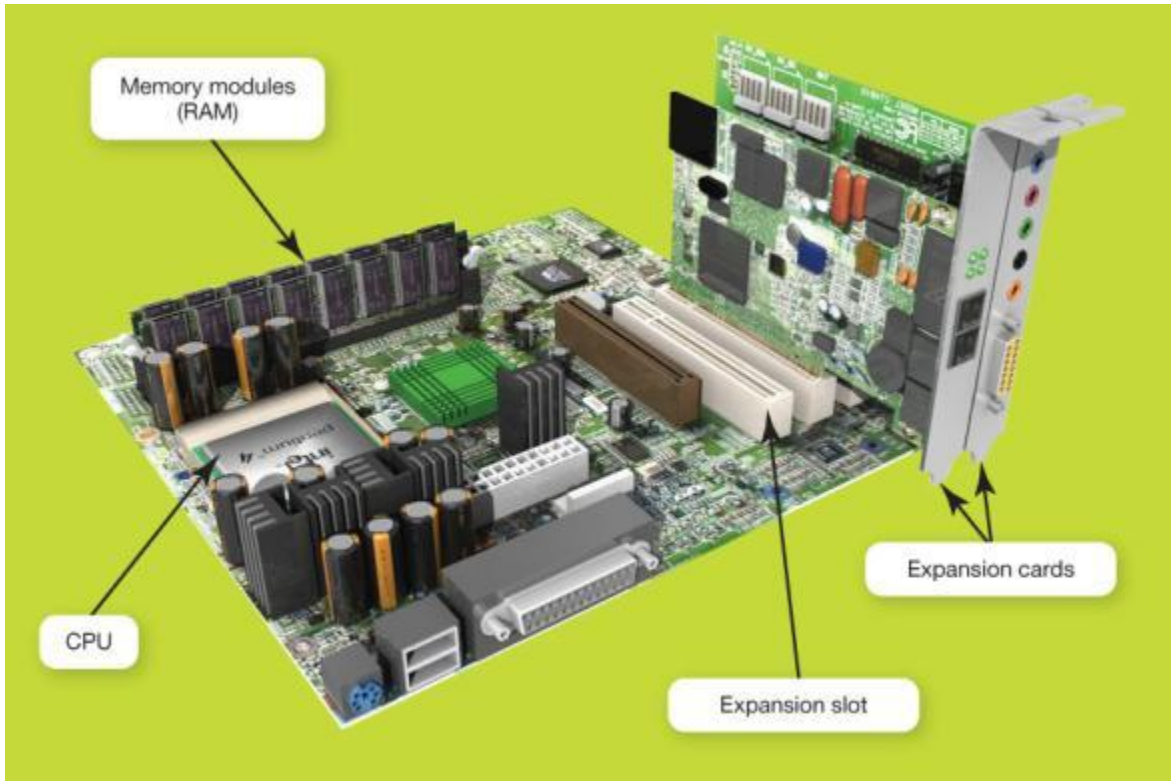
Computing Systems

- Hardware Components
  - Central Processing Unit (CPU)
  - Auxiliary Processors (GPU, etc)
  - Memory
  - Bus
  - Network Connection
  - External Devices: keyboard, monitor, printer
- Software Components
  - Operating System: Linux, MacOS, Windows, etc
  - Applications

Inside of a Desktop Computer

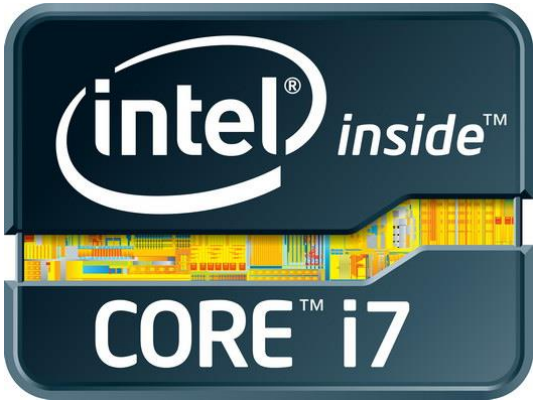
The Motherboard

- CPU
- RAM
- Expansion cards and slots
- Built-in components



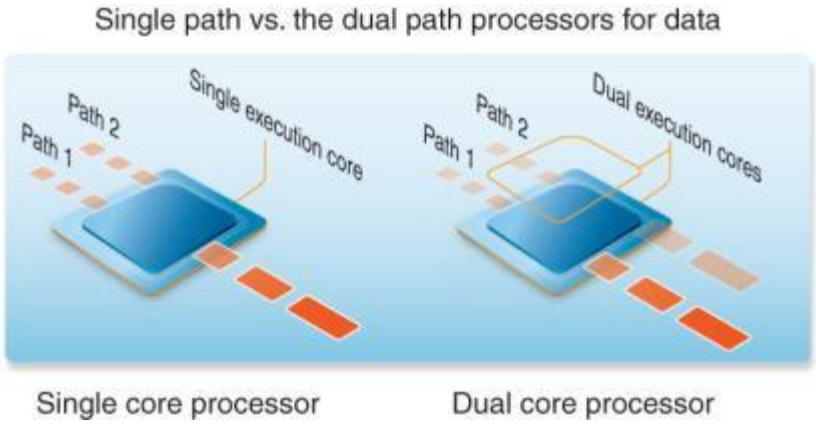
Central Processing Unit (CPU)

- Referred to as the “brains” of the computer
- Controls all functions of the computer
- Processes all commands and instructions
- Can perform billions of tasks per second



CPU Performance Measures

- Speed
  - Megahertz (MHz)
  - Gigahertz (GHz)
- Cores
  - Single
  - Dual
  - Quad
  - Eight
  - Hundreds?



Binary Numbers

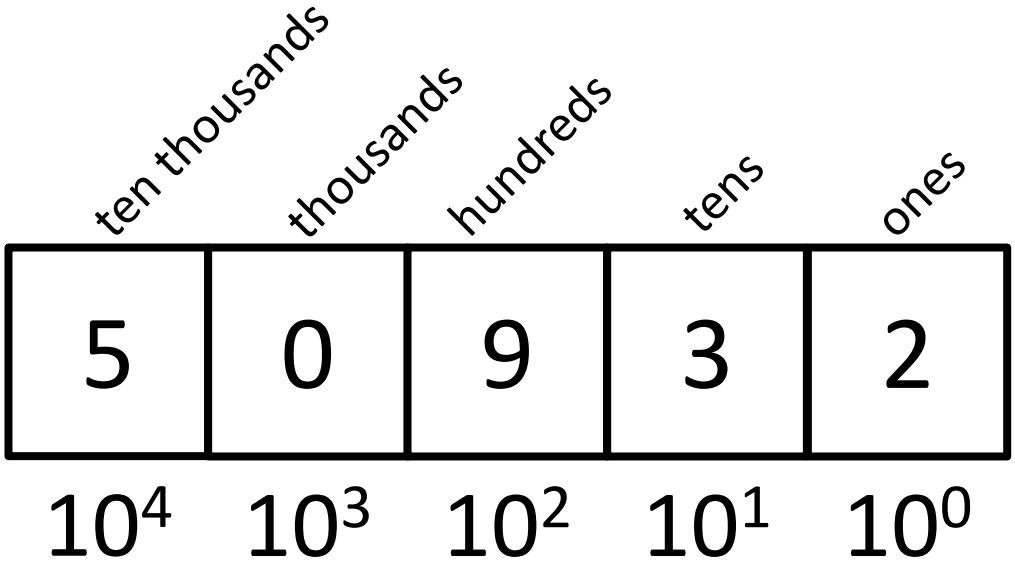
- Computers store all information (code, text, images, sound,) as a binary representation
  - "Binary" means only two parts: 0 and 1
- Specific formats for each file help the computer know what type of item/object it is
- But why use binary?

Decimal vs Binary

- Why do we use decimal numbers?
  - Ones, tens, hundreds, thousands, etc.
- But computers don't have fingers…
  - What do they have instead?
- They only have two states: "on" and "off"

Decimal Example

- How do we represent a number like 50,932?



$$
\begin{aligned}
2 \times 10^0 &= 2 \\
3 \times 10^1 &= 30 \\
9 \times 10^2 &= 900 \\
0 \times 10^3 &= 0000 \\
5 \times 10^4 &= 50000 \\
&\text{------} \\
\text{Total:} &\quad 50932
\end{aligned}
$$

Decimal uses 10 digits, so…

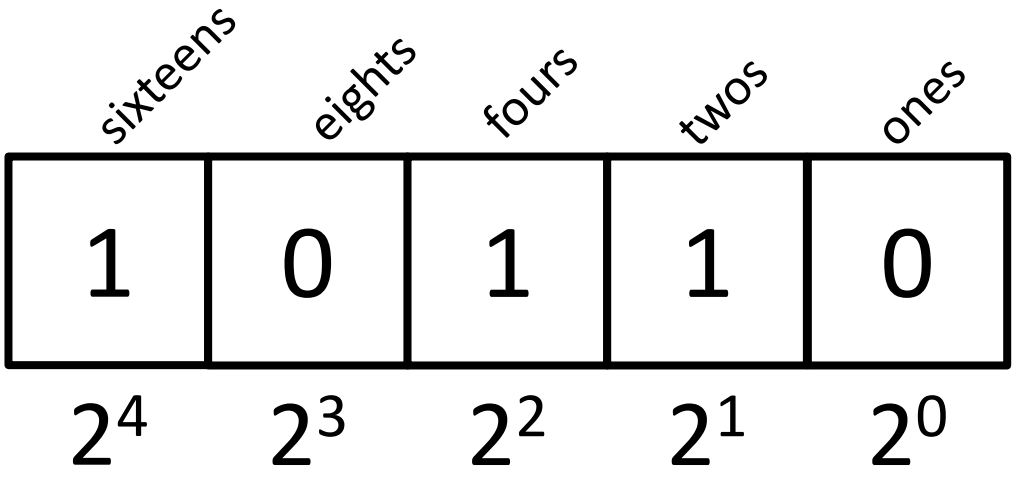
Decimal Example

6	7	4	9	3
10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1
60000	7000	400	90	3

60000+7000+400+90+3 = 67493

Binary Example

- Let's do the same with 10110 in binary



$0 \times 2^0 = 0$
$1 \times 2^1 = 2$
$1 \times 2^2 = 4$
$0 \times 2^3 = 0$
$1 \times 2^4 = 16$
--
Total: 22

Binary uses 2 digits, so our base isn't 10, but…

Binary to Decimal Conversion

- Step 1: Draw Conversion Box
- Step 2: Enter Binary Number
- Step 3: Multiply
- Step 4: Add

1	0	1	0	0	0	1	1	0	1
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
512	0	128	0	0	0	8	4	0	1

512+0+128+0+0+0+8+4+0+1 = 653

Decimal to Binary Conversion

- Step 1: Draw Conversion Box
- Step 2: Compare decimal to highest remaining binary.
- Step 3: If remainder is higher add 1 and subtract
- Step 4: Repeat until 0

Convert 643 to binary

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
1	0	1	0	0	0	0	0	1	1

643-512 = 131 131-128 = 3 3-2=1 1-2=0

Exercise: Binary to Decimal

- What are the decimals equivalents of…

```
101
1111
100000
101010
1000 0000
```

(Longer binary numbers are often broken into blocks of four digits for readability.)

Exercise: Binary to Decimal

- What are the decimals equivalents of…

```
101          = 1+0+4              = 5
1111         = 1+2+4+8            = 15
100000       = 0+0+0+0+0+32       = 32
101010       = 0+2+0+8+0+32       = 42
1000 0000    = 0+0+...+128        = 128
```

(Longer binary numbers are often broken into blocks of four digits for readability.)

Exercise: Decimal to Binary

- What are the binary equivalents of…

  9

  27

  68

  1000

Exercise: Decimal to Binary

- What are the binary equivalents of…

```
9    = 1001 (or 8+1)
27   = 0001 1011 (or 16+8+2+1)
68   = 0100 0100 (or 64+4)
1000 = 0011 1110 1000
       (or 512+256+128+64+32+8)
```

"Levels" of Languages

- Machine Code (lowest level)
  - Code that the computer can directly execute
  - Binary (0 or 1)
- Low Level Language
  - Interacts with the hardware of the computer
  - Assembly language
- High Level Language
  - Compiled or interpreted into machine code
  - Java, C++, Python

Compilation vs Interpretation

- Compiler
  - A complex computer program that takes another program and translates it into machine language
  - Compilation takes longer, but programs run faster
- Interpreter
  - Simulates a computer that can understand a high level language
  - Allows programming “on the fly”

Algorithmic Thinking

- Algorithms are an ordered set of clear steps that fully describes a process

- Examples from real life:
  - Recipes
  - Driving directions
  - Instruction manual (IKEA)

Exercise: PB&J Algorithm

- English speaking aliens are visiting Earth for the first time. They want to know how to make a peanut butter and jelly sandwich.

- Explicitly, what are the required steps for building a peanut butter and jelly sandwich?

Announcements

- No Labs for week of August 26th and 27th
- Make sure to log into the course Blackboard
  - Let us know if you have any problems
- Course website will be announced when it is completed