

# Lecture 1: Course overview

David Hovemeyer

August 30, 2021

601.229 Computer Systems Fundamentals



# Welcome!

- ▶ Welcome to CSF!
- ▶ Today:
  - ▶ Administrative stuff
  - ▶ Course overview
  - ▶ Binary data representation

# Administrative stuff

# About the course

- ▶ Instructor
  - ▶ David Hovemeyer, [daveho@cs.jhu.edu](mailto:daveho@cs.jhu.edu), Malone 240A
- ▶ CAs
  - ▶ Coming soon, see course web page for details

# Where to find stuff

- ▶ Course website: <https://jhucsf.github.io/fall2021>
  - ▶ Syllabus, schedule, lecture notes, assignments, etc.
  - ▶ All public course information will be here
- ▶ Campuswire <https://campuswire.com>
  - ▶ Non-public course information such as homework/exam solutions
  - ▶ Discussion forum, Q/A: please post questions here!

# Syllabus highlights

- ▶ Please read the syllabus carefully:  
<https://jhucsf.github.io/fall2021/syllabus.html>
- ▶ Highlights:
  - ▶ Grades: 55% homework, 40% exams, 5% participation
  - ▶ 6 or 7 assignments, mostly programming based, expect them to be challenging!
  - ▶ Late policy: you have 120 late hours to use as needed (assignment submissions which exceed the late day limit receive no credit)
  - ▶ Three exams (two during semester, one during final exam period)
    - ▶ Exams will be in-class
    - ▶ Location will be announced in class and posted on Campuswire (we will use a large lecture hall to permit social distancing)
    - ▶ Will focus on recently-covered material

# Academic integrity

- ▶ Please read the academic integrity policy in the syllabus carefully
- ▶ Highlights:
  - ▶ Follow the CS Academic Integrity Code:  
<https://www.cs.jhu.edu/academic-integrity-code/>
  - ▶ Homework assignments
    - ▶ Individual: code sharing is not allowed
    - ▶ Pair: you can work with one partner
  - ▶ Exams are (obviously) individual effort
  - ▶ Violations of academic integrity will be reported to the Student Conduct office
- ▶ Be careful about using web as a resource
  - ▶ Do *not* copy code
  - ▶ *Always* cite sources used

# Class meetings

- ▶ Typical class meeting: lecture/discussion, peer instruction questions, occasional group activities, discussion of current assignment, time for free-form Q&A
- ▶ *Do the reading in advance!*
- ▶ Come prepared to actively engage with the material!
  - ▶ Learning is not passive
  - ▶ More productive class time → better outcomes
  - ▶ Ask questions!



# Peer instruction

- ▶ How peer instruction works:
  - ▶ Slide with a multiple choice question
  - ▶ Answer individually, discuss with peers, then answer again
  - ▶ Shown to improve outcomes!
  - ▶ Questions may be challenging
  - ▶ Graded for participation only
- ▶ You may have done this in other courses

# Peer instruction etiquette

- ▶ Discussion groups for peer instruction will be assigned randomly (using Zoom breakout rooms)
- ▶ Be respectful:
  - ▶ Let everyone participate
  - ▶ Don't put down anyone else's ideas
- ▶ Work together and think carefully about the question!

# First clicker quiz!

Clicker quiz omitted from public slides

# Computing requirements

- ▶ All assignments will be done using x86-64 Linux
- ▶ Autograders will use Ubuntu 18.04
- ▶ Doing your development and testing on ugrad machines is *generally* fine (but note that compiler and other tools will likely be different than Gradescope)
- ▶ Development on Windows or MacOS is not recommended
  - ▶ Except: Windows Subsystem for Linux is an excellent option!
- ▶ Running Ubuntu 18.04 in a VirtualBox VM highly recommended: see Resources page on course website
- ▶ *Get your development environment set up NOW*
- ▶ If you are using an M1 Mac, a virtual machine isn't an option (you need an x86-64 CPU)

# Course overview

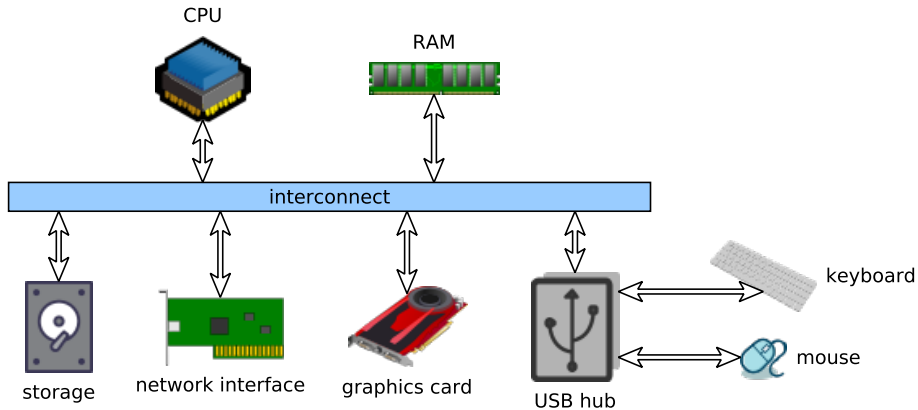
# What the course is about

- ▶ Course is about *computer systems* from the *programmer's perspective*
- ▶ Computer system = hardware + software
  - ▶ Much of our concern is the interaction between hardware and software — how they work together

# Goals of course

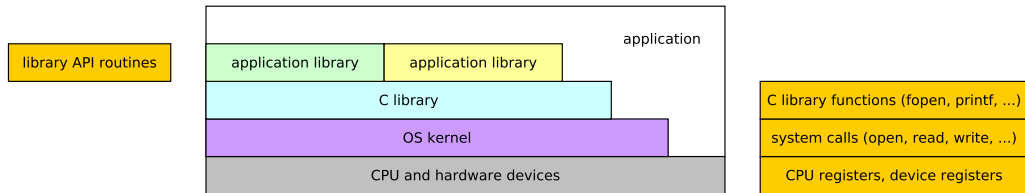
- ▶ “Deep” understanding of how computers work (down to hardware)
  - ▶ OS and runtime library interfaces
  - ▶ Machine-level ISA / assembly language
  - ▶ Processor features
  - ▶ Operating system features
- ▶ Apply this understanding to...
  - ▶ Optimize application performance
  - ▶ Avoid pitfalls such as security vulnerabilities
  - ▶ Take full advantage of the computer's and operating system's capabilities

# A computer system (hardware)



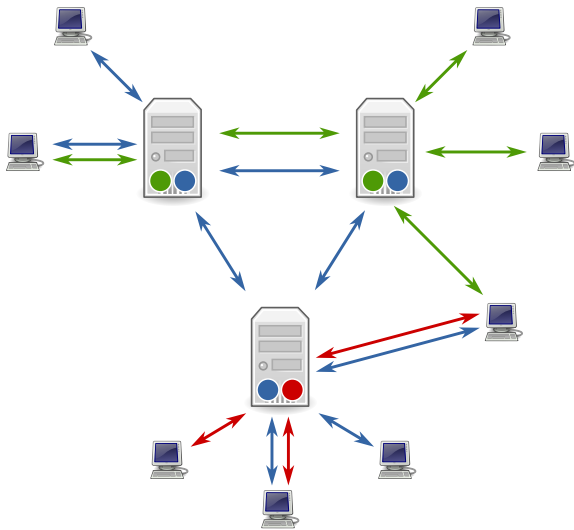


# A computer system (software)



- ▶ Your application program is supported by lower layers of software and hardware
- ▶ Each layer provides an interface to the layer above

# A computer network



Computer networks allow your program to communicate with peer systems.

Thanks to the global Internet, the peer systems could be anywhere on earth!

# Binary data representation

# Discrete data representation

- ▶ Digital computers use a *discrete* representation for all data
- ▶ Consider a representation of a number:

# Discrete data representation

- ▶ Digital computers use a *discrete* representation for all data
- ▶ Consider a representation of a number:
  - ▶ A *continuous* representation would allow the number to have *any* value
    - ▶ We think of physical phenomena (mass, velocity, etc.) as being continuous

# Discrete data representation

- ▶ Digital computers use a *discrete* representation for all data
- ▶ Consider a representation of a number:
  - ▶ A *continuous* representation would allow the number to have *any* value
    - ▶ We think of physical phenomena (mass, velocity, etc.) as being continuous
  - ▶ A *discrete* representation would allow the number to have one of a set of possible values, where the set of possible values is *enumerable*
    - ▶ Often we think of discrete values as corresponding to a range of integers

# Why discrete representation?

- ▶ Why do digital computers use a discrete representation for all data?

# Why discrete representation?

- ▶ Why do digital computers use a discrete representation for all data?
- ▶ Answer: internally, information is represented using *digital voltages*



# Why discrete representation?

- ▶ Why do digital computers use a discrete representation for all data?
- ▶ Answer: internally, information is represented using *digital voltages*
  - ▶ High voltage (1) vs. low voltage (0)
  - ▶ Digital circuits (with discrete high vs. low voltages) have many advantages over *analog* circuits, where voltages can vary continuously

# Why discrete representation?

- ▶ Why do digital computers use a discrete representation for all data?
- ▶ Answer: internally, information is represented using *digital voltages*
  - ▶ High voltage (1) vs. low voltage (0)
  - ▶ Digital circuits (with discrete high vs. low voltages) have many advantages over *analog* circuits, where voltages can vary continuously
- ▶ OK, let's think about what discrete data representations will look like...

# Why discrete representation?

- ▶ Why do digital computers use a discrete representation for all data?
- ▶ Answer: internally, information is represented using *digital voltages*
  - ▶ High voltage (1) vs. low voltage (0)
  - ▶ Digital circuits (with discrete high vs. low voltages) have many advantages over *analog* circuits, where voltages can vary continuously
- ▶ OK, let's think about what discrete data representations will look like...
  - ▶ Starting with *integers* (if you can represent integers, you can represent anything)

# Decimal numbers

- ▶ We're all familiar with decimal (base 10) numbers
- ▶ E.g.,

$$42 = 4 \cdot 10^1 + 2 \cdot 10^0$$

- ▶ Digits are 0–9
- ▶ Places are powers of 10

# Other bases

- ▶ Base 10 is arbitrary!
- ▶ Representing decimal 42 using base 5:

$$42_{10} = 132_5 = 1 \cdot 5^2 + 3 \cdot 5^1 + 2 \cdot 5^0$$

- ▶ “Digits” are 0–4
- ▶ Places are powers of 5

# Try it!

How to express decimal 42 using base 6?

$$\underline{\quad} \cdot 6^2 + \underline{\quad} \cdot 6^1 + \underline{\quad} \cdot 6^0$$

How to express decimal 79 using base 6?

$$\underline{\quad} \cdot 6^2 + \underline{\quad} \cdot 6^1 + \underline{\quad} \cdot 6^0$$

Reference:

$$6^2 = 36$$

$$6^1 = 6$$

$$6^0 = 1$$

# Binary

- ▶ Binary = base 2
- ▶ Representing decimal 42 using base 5:

$$\begin{aligned}42_{10} &= 101010_2 \\ &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0\end{aligned}$$

- ▶ “Digits” are 0 and 1
- ▶ Places are powers of 2
- ▶ Computers use binary representations for all data, because
  - ▶ *Digital circuits* use two voltage levels, high and low
  - ▶ By convention, 1=high voltage, 0=low voltage
  - ▶ So, computer hardware fundamentally operates on binary data

# Try it!

How to express decimal 29 using base 2?

$$\underline{\quad} \cdot 2^5 + \underline{\quad} \cdot 2^4 + \underline{\quad} \cdot 2^3 + \underline{\quad} \cdot 2^2 + \underline{\quad} \cdot 2^1 + \underline{\quad} \cdot 2^0$$

Reference:

$$2^5 = 32$$

$$2^4 = 16$$

$$2^3 = 8$$

$$2^2 = 4$$

$$2^1 = 2$$

$$2^0 = 1$$