# CS 220 Course Overview

Dr. Kai Presler-Marshall (“Dr. Kai”)

# GETTING ACQUAINTED WITH CS 220

# Course Overview

- Who?
- Where?
- When?
- What?
- Why?
- How?

# Course Overview: Who?

- Professor: Dr. Kai
  - I’m not really that picky on what you call me, as long as it’s polite
  - Professor Presler-Marshall is fine too, just verbose ☺
- Office is in Malone 337
  - Feel free to drop by any time the door is open
- Email: kai@cs.jhu.edu

# Course Overview: Who?

- TAs:
  - About a dozen, shared with other sections of IP
  - Office hours TBD, but will be spread out throughout week

# Course Overview: Where?

- IRL:
  - 310 Maryland Hall
- Online:
  - Course materials: https://jhu-ip.github.io/cs220-sp26/
  - Q&A: https://piazza.com/jhu/spring2026/en601220/home
  - Assignment submission: https://www.gradescope.com/
  - We do have a Canvas page as well, but won’t be using it

# Course Overview: When?

- Mondays, Wednesdays, & Friday
  - 10 -> 11:15 AM (Section 1)
  - 12 -> 1:15 PM (Section 2)
- Y’all made it here this morning, so that’s good
- Attendance is expected, and will be checked by the TAs
  - Why? Showing up to class encourages regular engagement with material, which is associated with you learning more!

# Course Overview: What?

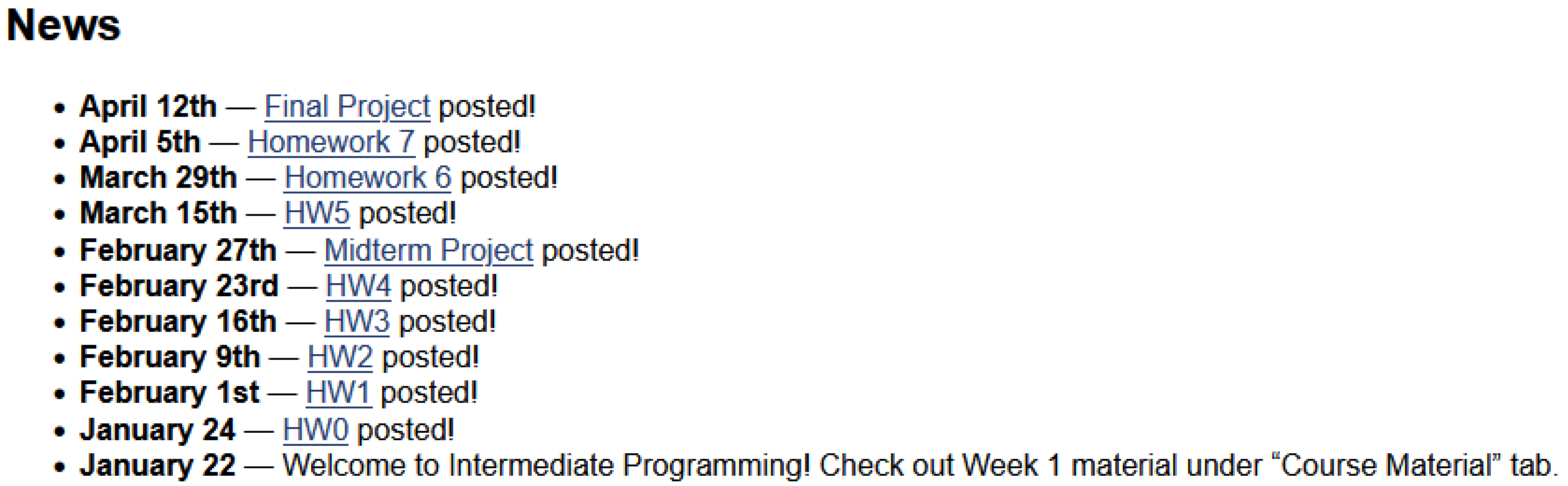
- Intermediate Programming teaches you to solve intermediate-to-advanced problems, using C & C++
- We’ll discuss “Why these languages?” in a moment

# Course Overview: What?

- Your Grade:
  - 4% - Attendance & Participation
  - 6% - handwritten homeworks (3)
  - 14% - coding homeworks (4 big ones, 1 small one)
  - 7% - midterm coding project (in teams of two)
  - 30% - midterm exam (in class)
  - 7% - final coding project (in teams)
  - 30% - final exam (in class)
  - 2% - fudge grade (see details on the website)
- Everything except the exams will be submitted on Gradescope
- See the syllabus ("Expectations & Grading" and "Homework Policy") for details

# Course Overview: What?

- Homeworks will be announced on the main class page
  - These dates, obviously, are not this semester ;)



## News

- **April 12th** — Final Project posted!
- **April 5th** — Homework 7 posted!
- **March 29th** — Homework 6 posted!
- **March 15th** — HW5 posted!
- **February 27th** — Midterm Project posted!
- **February 23rd** — HW4 posted!
- **February 16th** — HW3 posted!
- **February 9th** — HW2 posted!
- **February 1st** — HW1 posted!
- **January 24** — HW0 posted!
- **January 22** — Welcome to Intermediate Programming! Check out Week 1 material under “Course Material” tab.

# Course Overview: Why?

- By this point, you should have some experience with Python or Java
  - And if you’re taking Data Structures as well, you’ll be getting a lot more Java there
- So, why learn C & C++?
- As I see it, there are two main reasons for computer scientists to learn new languages
  - Different languages are suited to solving different types of problems
  - Different languages encourage you to think about problems in different ways

# Course Overview: Why?

- Different languages have different strengths
  - Java is a horrendous choice for doing data science or machine learning
  - You’d probably not want to build a high-performance web application in Python
- So, what are C/C++ good for?
- C & C++ are languages well-suited for writing *low-level* code that needs to interact more closely with underlying hardware
- Or, for platforms where the overhead of the JVM or Python interpreter is unbearable
- If you ever end up doing *embedded programming* or writing *device drivers*, C or C++ is the natural choice

# Course Overview: Why?

- I like both Python and Java tremendously
  - Python’s minimal syntax makes it fantastic for prototyping – trying out an idea to see if it’s viable
  - Java’s more verbose syntax and compile-time type checks make it easier to pick up some code I (or someone else) wrote previously and figure out, “What is this doing?”
- Both Python and Java are somewhat magical
  - How does Python evaluate: `"cat" in "catherine"` or `my_list[::2]`?
  - When you’re done with objects in Java, you throw them on the ground, and the JVM sweeps them up. How?

# Course Overview: Why?

- From a *mechanics* standpoint, C is easily the simplest language I know
- There is no magic in C. Everything* that you want done, you have to do yourself
- Learning C teaches us how the things that we take for granted in other programming languages actually work
- At the same time, we'll still see how C can be used to solve non-trivial problems
- C++ builds on this – we'll see how C++ is an evolution of C, and how it serves as a midpoint between C and f.ex Java

# Course Overview: Why?

- There are many different *paradigms* by which you can classify programming languages
  - Java is a statically-typed language that forces you to think in an object-oriented paradigm
  - Python is a duck-typed language that supports both functional and object-oriented programming
  - Clojure is a *functional* programming language
- Learning different paradigms broadens your problem-solving expertise
  - No longer do you have to try and force the same solution on every problem

# Course Overview: Why?

- C & C++ teach you different ways of looking at problems
  - C does not support OOP
    - You can fake it (and I've done so), but it's not fun
    - Instead, C is a *procedural* programming language, where you solve problems by breaking them into functions, not objects
  - In C++, there are no nulls (unless you use C-style pointers), and objects are copied-on-assignment
    - In Java, you either need to defensively perform null checks, or your code will explode
    - C++ avoids this
    - Combined with copy-on-assign, it makes you put more thought into object ownership, which can encourage better programming habits
      - We'll talk about what this actually means later in the semester

# Course Overview: How?

- Intermediate Programming is taught as a *flipped class*
- You’ll have videos to watch **before** class that cover the new material for the day
  - Usually 2, sometimes 1, sometimes 3
- In class, we will:
  - Do a brief recap of the activity from the previous day
  - Do a brief recap of the videos for the day
  - Start on a new exercise
    - We might not always finish in class – if so, you’ll finish after class
    - These are turned in on Gradescope and count towards your participation score

# What Does Success Look Like?

- Do all of the homeworks on time
  - This includes *starting early* too – don’t wait until Wednesday to start a homework due Thursday
  - This also applies to projects
- Show up to class, and participate actively in the activities – don’t just sit here
  - These in-class activities are worth a few points to encourage you to complete them
- Keep at it when things get hard – learning is hard, but this process of struggling & overcoming your difficulties is how you learn
- Solve problems yourself
  - Using the book, reference materials provided, etc is fine – but don’t use Chegg, ChatGPT, etc
- Think through problems logically instead of just hacking at it
  - What I mean is, don’t just try stuff at random & hope it works
  - This won’t work on the exam, since you can’t run any code

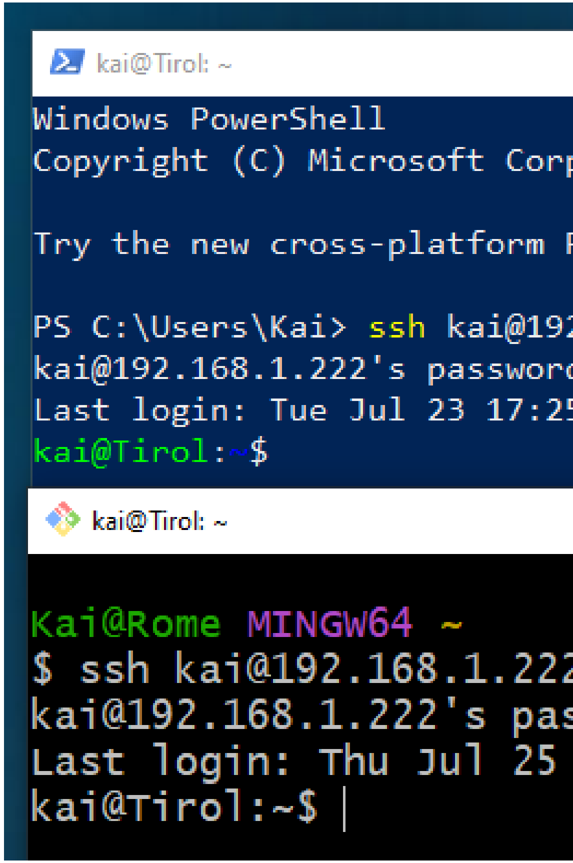
# CS LINUX ACCOUNTS

# Computing Environments

- One of the objectives of this course is for you to learn more about command-line Unix environments & tools
- These environments and tools are *very helpful* for computer scientists
  - Many very helpful tools are command-line only
  - Command-line environments can easily be scripted, allowing you to automate tedious/repetitive tasks
- To provide a consistent environment, we'll be using the CS department's Linux system for this course
  - Details at the end of this slide deck

# Connecting to Linux

- While Linux does support point-and-click (graphical) user interfaces, we'll be using the command-line interface
- To connect to a Linux system, we use `ssh` (**s**ecure **sh**ell)
  - `ssh username@hostname`
    - `ssh` [kai@192.168.1.222](kai@192.168.1.222) (to connect to a system on my home network)
    - `ssh` [kai@ugradx.cs.jhu.edu](kai@ugradx.cs.jhu.edu) (to connect to the JHU CS environment)
- If you're on Linux already open up a Terminal, then run the `ssh` command above
  - Same deal on MacOS
- On Windows, you can do one of the following:
  - Download [PuTTY](PuTTY) and connect from there
  - Open PowerShell (not Command Prompt) and run `ssh` command
  - Download & install [Git Bash](Git Bash), then run `ssh` command
    - We'll discuss Git on Friday – it's one of the awesome tools we'll learn this semester – so this will come in handy then

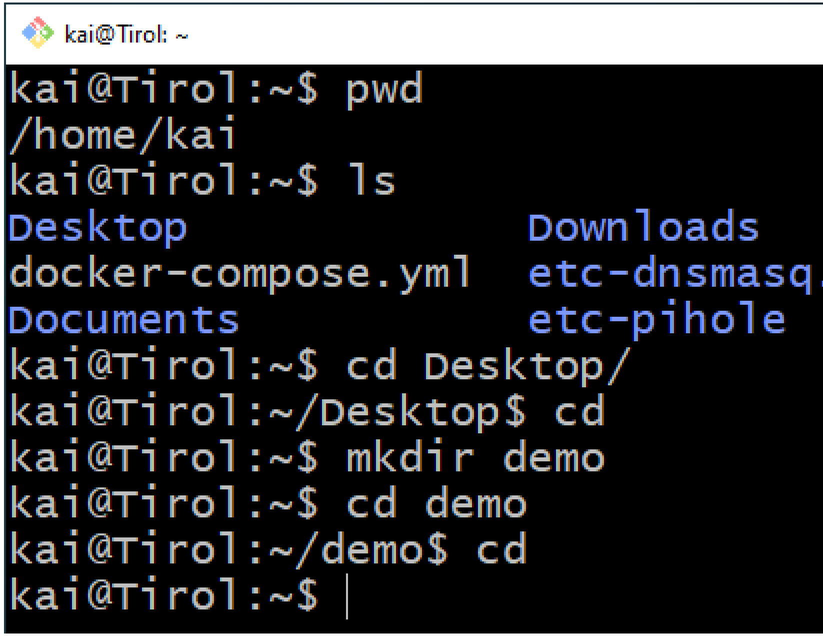


```
Windows PowerShell
Copyright (C) Microsoft Corp

Try the new cross-platform P

PS C:\Users\Kai> ssh kai@192
kai@192.168.1.222's passwor
Last login: Tue Jul 23 17:2
kai@Tirol:~$
```

```
Kai@Rome MINGW64 ~
$ ssh kai@192.168.1.222
kai@192.168.1.222's pas
Last login: Thu Jul 25
kai@Tirol:~$
```

# Linux Basics

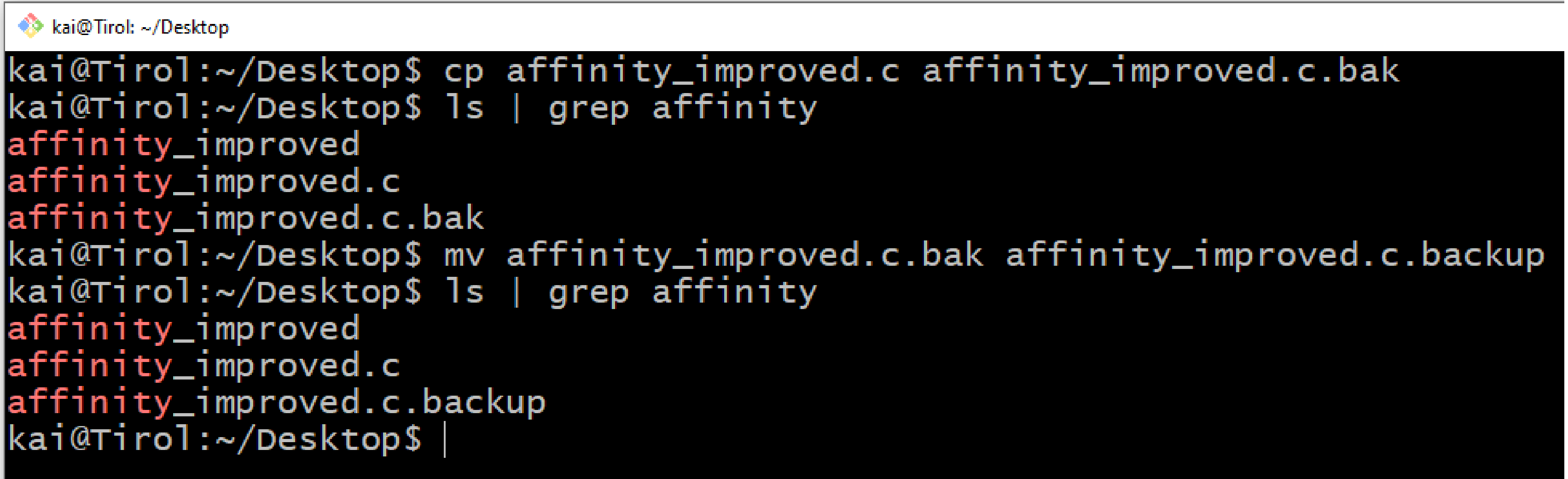
- `pwd` (**p**rint **w**orking **d**irectory) – where am I?
- `ls` (**l**i**s**t) – list files/folders
  - `-l` creates a tabular list output
  - `-a` shows hidden files/folders
  - `-la` does both
- `cd` (**c**hange **d**irectory) – go somewhere
  - `cd folder` enters that folder
  - `cd` goes home (`/home/<you>`)
  - Paths can be absolute (starting with /) or relative to current directory (no /)
    - This applies to *all* of these commands
- `mkdir` (**mak**e **dir**ectory) – create new folder
  - `-p` lets you create multiple directories in one command, f.ex `mkdir -p first/second/third` creates all three



```
kai@Tirol:~$ pwd
/home/kai
kai@Tirol:~$ ls
Desktop             Downloads
docker-compose.yml  etc-dnsmasq
Documents           etc-pihole
kai@Tirol:~$ cd Desktop/
kai@Tirol:~/Desktop$ cd
kai@Tirol:~$ mkdir demo
kai@Tirol:~$ cd demo
kai@Tirol:~/demo$ cd
kai@Tirol:~$
```

# Linux Basics

- `less <filename>` – lets you see contents of a file, one screen at once
  - F.ex `less my_file.c`
- `mv src dest` **(move)** – moves files or folders
  - `mv my_file.c ex1.c` renames file
- `cp src dest` **(copy)** – copies files or folders
  - `cp ex1.c ex1.c.bak` – creates a second, backup, copy
- `grep` – search for matching text
- More details for all of these files available on the man pages – f.ex search “man cp” & click on link to https://man7.org/linux/man-pages/man1/cp.1.html



```
kai@Tirol:~/Desktop$ cp affinity_improved.c affinity_improved.c.bak
kai@Tirol:~/Desktop$ ls | grep affinity
affinity_improved
affinity_improved.c
affinity_improved.c.bak
kai@Tirol:~/Desktop$ mv affinity_improved.c.bak affinity_improved.c.backup
kai@Tirol:~/Desktop$ ls | grep affinity
affinity_improved
affinity_improved.c
affinity_improved.c.backup
kai@Tirol:~/Desktop$ |
```