Agenda for Today

1. Lab Q&A
2. Discussion: UNIX and *Worse is Better*
3. Closing Thoughts
4. End-of-Course Evaluations
Q&A
Do you recommend creating a struct to store information about peers?
That might be a good idea, although I’ve seen solutions that just use a file descriptor for each connection.
Some kind of synchronization is required, since you could have one thread adding to the list while another is traversing it.

*Warning:* you can’t assume all messages from a particular peer will have the same username.
How does peer-to-peer networking impact network usage?

The capacity we need from the network at each peer’s connection is proportional to the number of peers it is connected to. You come up with schemes for organization connections that never require huge amounts of bandwidth. Spreading connections out over many peers means that one peer won’t be overloaded.
P2P Chat Lab Q&A
Discussion: UNIX and Worse is Better
Basic Concepts

What is the MIT approach?

We worry about completeness and making code correct more than how difficult it is to finish. Implementation simplicity is less important than correctness.
Examples: Scheme and Common Lisp

What is the New Jersey approach?

We want systems to be mostly complete and mostly correct, but it’s more important that it is simple. We care about correctness mostly for the observable parts of the system. We want both the implementation and the interface to the system to be simple, although we place higher priority on implementation simplicity.

Examples: C and UNIX
Worse is Better: Understanding the Author

Why does Gabriel refer to C and UNIX as "the ultimate computer viruses?"

C and UNIX spread really easily. They work well enough, and they work well on any kind of machine. It’s easy to adapt them to whatever kind of machine you have. RPG passes a bit of judgement by calling them “viruses” to make that point that C and UNIX are “bad”.

Which approach do you think Gabriel prefers? Why does he prefer it?

It seems like RPG likes the NJ approach better because more people end up using your work. Once a systems spreads, more people will care about making it better.
Worse is Better: Preferences

Which approach do you prefer? Why?

NJ: Using the MIT approach feels right, but often leaves you stuck with a system that doesn’t work at all. The NJ approach makes it easier to break up hard problems.
NJ: Implementing and changing our work is easier. The MIT approach is less flexible.
NJ: In a classroom setting, it’s nice to learn from mistakes.
NJ: This sounds like how startups work, and maybe make money.
NJ: “Implementing the last 20% takes 80% of the time”
NJ: This approach fits with the way open source software works.

Reading recommendation: *The Cathedral and the Bazaar* by Eric S. Raymond

MIT: Doing the perfect thing just works better. Specialized systems might justify that.
MIT: CUDA is perfect for working with Nvidia GPUs, for example.
MIT: Some use cases demand perfection. Safety-critical systems, for example.
MIT: Planning out a program in advance might save a lot of implementation and debugging time. But sometimes we don’t know how to solve problems AOT.
Takeaways

Does either reading change how you think about building systems? How?

- I was hesitant to jump onboard with the NJ approach because I am uncomfortable delivering incomplete work. But I do see the advantages of the NJ approach.
- I wasn’t strongly swayed, but I was entertained. I feel validated knowing that trying things and possibly failing or ending up with a partial solution is okay.
- I tend to focus on using the MIT approach, but after reading this I may try the NJ approach.
- I have mixed feelings. I still plan to approach work for school with an MIT approach.
- I will think about code and solutions in an MIT way, but my actual work may be more NJ-inflected. It’s dangerous to focus on just one approach or the other.
- I used to take more of a NJ approach, but I think interfaces should be simple so NJ may not be a perfect fit for my thinking. I still care about correctness.
- I used to follow the MIT approach, but I think I will shift to a NJ approach as work in CS courses grows more and more difficult.
Motivation

Why do you think I assigned these readings for the end of this class?
Closing Thoughts
What I hope you take away from CSC 213

CS Concepts
As in any CS course, we learn about new CS concepts or explore concepts you've seen before in greater depth.

Problem Solving Strategies
Writing large programs (especially in C) requires a lot of extra work; doing this for labs and your projects helps you practice new techniques for avoiding or solving problems you may encounter.

Other Skills and Experience
As in any course, there are other skills we practice beyond the core technical content of the class.
CS Concepts

What CS concepts did we practice in this course? Why did we include each one?
CS Concepts

What CS concepts did we practice in this course? Why did we include each one?

**CPU Scheduling** - This is at the “heart” of the system. Being efficient is important.

**Dealing with Memory** - Things can go very wrong if we don’t do this right! It’s hard.

**Concurrency** - Working with threads and GPUs was interesting. These have practical applications. Identifying situations where concurrency helps is tricky, and the coming up with implementations that work is really hard.

**Interacting with the OS** - How can we make the OS help us when we write programs?

**Networks** - In the “real world” almost everything uses networks/the internet. We need to know how to operate in that environment.

**System Design** - Think about why things work the way they do, and how to use them. Practice designing approaches to problems we haven’t seen before.
Problem Solving Strategies

What problem solving strategies did you learn or practice in this course?
Problem Solving Strategies

What problem solving strategies did you learn or practice in this course?
Other Skills & Experience

What general skills or experiences did you get from this course?
Why is each one important?

**How to work collaboratively** - We’ll have to do this in our careers. Hopefully you also found that working together on tough problems gives you an advantage.

**Thinking about where things can go wrong** - Learning how to spot and solve problems prepared us to find subtle bugs.

**Be modest about your own understanding** - My expectations for how something will work are often wrong. I may not be an expert, even when I feel pretty confident. Check your assumptions.

**Learn how to struggle/suffer with a problem** - The goal is not to be miserable, but when you’re working on difficult problems you’ll spend most of your time not knowing how to solve it (or discovering that you didn’t actually know how to solve it). Learning to function comfortable in that environment is important.
Other Skills & Experience

What general skills or experiences did you get from this course? Why is each one important?
End-of-Course Evaluations
End-of-Course Evaluations

Think carefully about your biases

We know these evaluations yield some biased outcomes, especially for groups underrepresented in the sciences. Make sure your scores and comments are based on your experience in the class, not your preconceptions.

Write comments

I read every comment (after grades are turned in) and use them to help refine the class. Tell me what worked, what didn't, and why.

Complete evaluations soon

I'm giving you time now, so hopefully you'll finish it now. If you need more time to think first, make sure you turn in the evaluation by Wednesday.
Thank You!