CS111, Lecture 27
Wrap-Up / What's Next?

Plan For Today

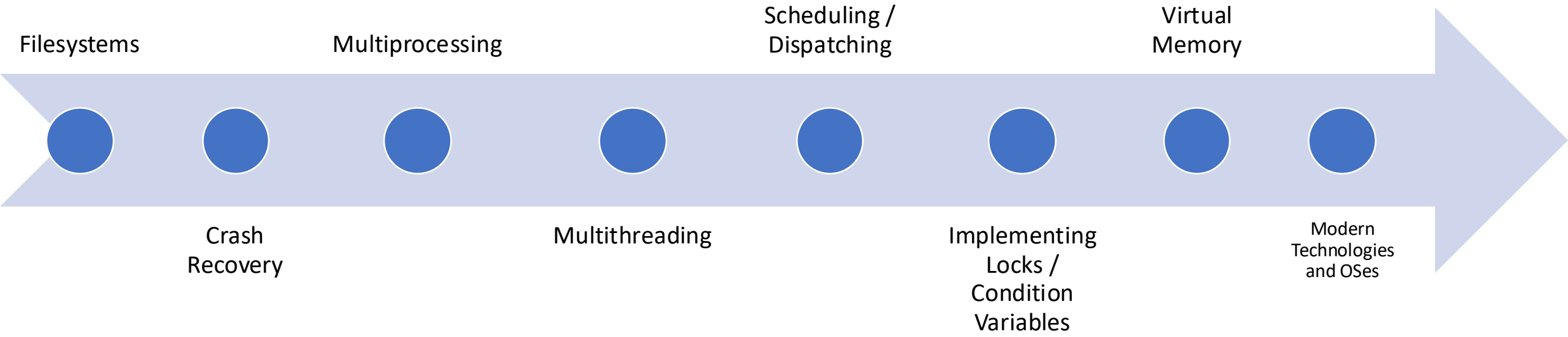
- **Recap**: Where We've Been
- Big Ideas
- What's Next?
- Q&A

Plan For Today

- **Recap: Where We've Been**
- Big Ideas
- What's Next?
- Q&A

We've covered a *lot* in just 10 weeks! Let's take a look back.

Our CS111 Journey



Course Overview

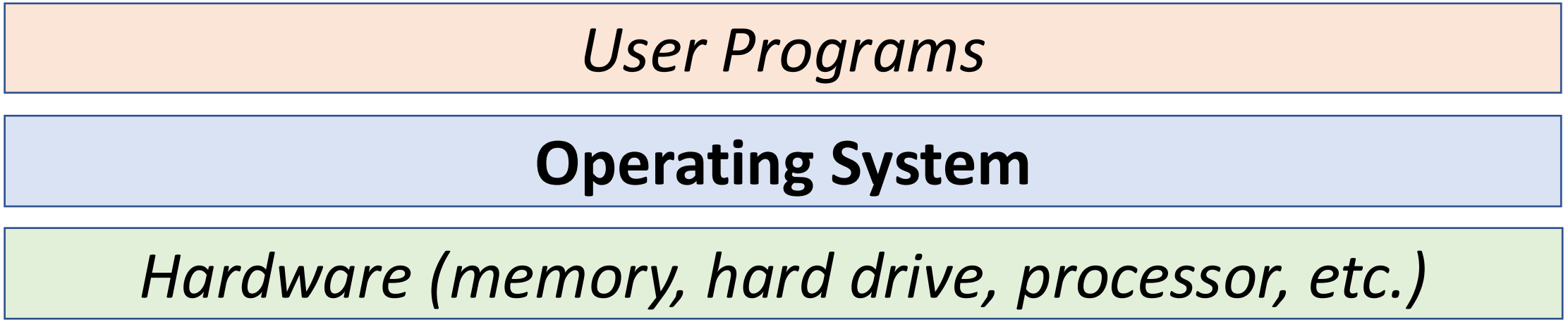
1. **Filesystems** - *How can we design filesystems to manage files on disk, and what are the tradeoffs inherent in designing them? How can we interact with the filesystem in our programs?*
2. **Multiprocessing** - *How can our program create and interact with other programs? How does the operating system manage processes?*
3. **Multithreading** - *How can we have concurrency within a single process? How does the operating system support this?*
4. **Virtual Memory** - *How can one set of memory be shared among several processes? How can the operating system manage access to a limited amount of system memory?*
5. **Modern Technologies and OSes** - *How do hardware advances impact the design of operating systems?*

First Day: What is an Operating System?

The operating system sits between the hardware and user programs. It manages shared resources and provides functionality for programs to run.

It manages things like:

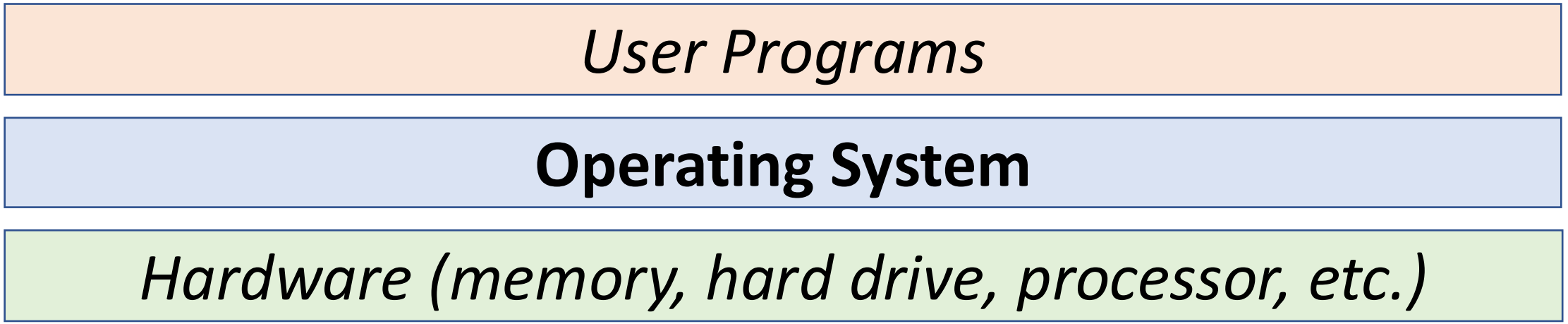
- Processor (CPU): decides what program gets to do work and for how long
- Memory (RAM): decides what programs get to use what areas of memory
- Hard Drive: decides how the disk is used to store files



First Day: What is CS111?

In CS111 we are going to explore both “sides” of operating systems:

- We’ll learn what functionality is exported by operating systems to make the programs that we write more powerful.
- We’ll learn how the operating system provides that functionality and how it acts as an **interface** to the computer hardware.



Operating Systems

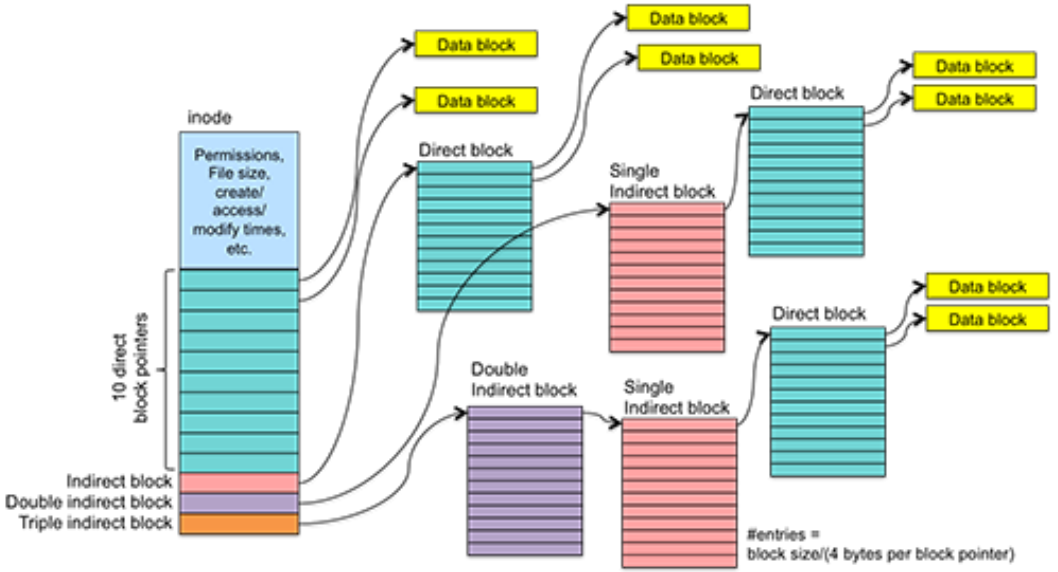
Why is it useful to know about operating systems?

- Understanding computing at this level demystifies how these seemingly-complex systems work and can aid future projects you work on.
- OSes contain many examples of elegant ideas in computing (concurrency, virtualization) that apply well beyond OSes, and pull together ideas like data structures, algorithms, languages, etc.
- We can learn how we can maximally take advantage of the hardware and operating system software available to us in our programs.
- Operating Systems are constantly evolving and encountering new applications (e.g., large datacenters) and new challenges

Filesystems

Key Question: *How can we design filesystems to manage files on disk, and what are the tradeoffs inherent in designing them? How can we interact with the filesystem in our programs?*

- Various design approaches such as contiguous allocation, linked files, FAT, and multi-level indexes, each with tradeoffs around fragmentation, file access, and amount of metadata
- Crash recovery adds additional questions
- We can use file descriptors to read/write files in our own programs.



Unix Filesystem Inode Design [source]

Filesystems

Why does this matter?

- Great example of the challenges and tradeoffs in designing large systems
- Shows us how we can directly manipulate files in our programs and what really happens when we open a file.
- Glimpse at hardware challenges/limitations – hard drives vs. flash storage

Multiprocessing

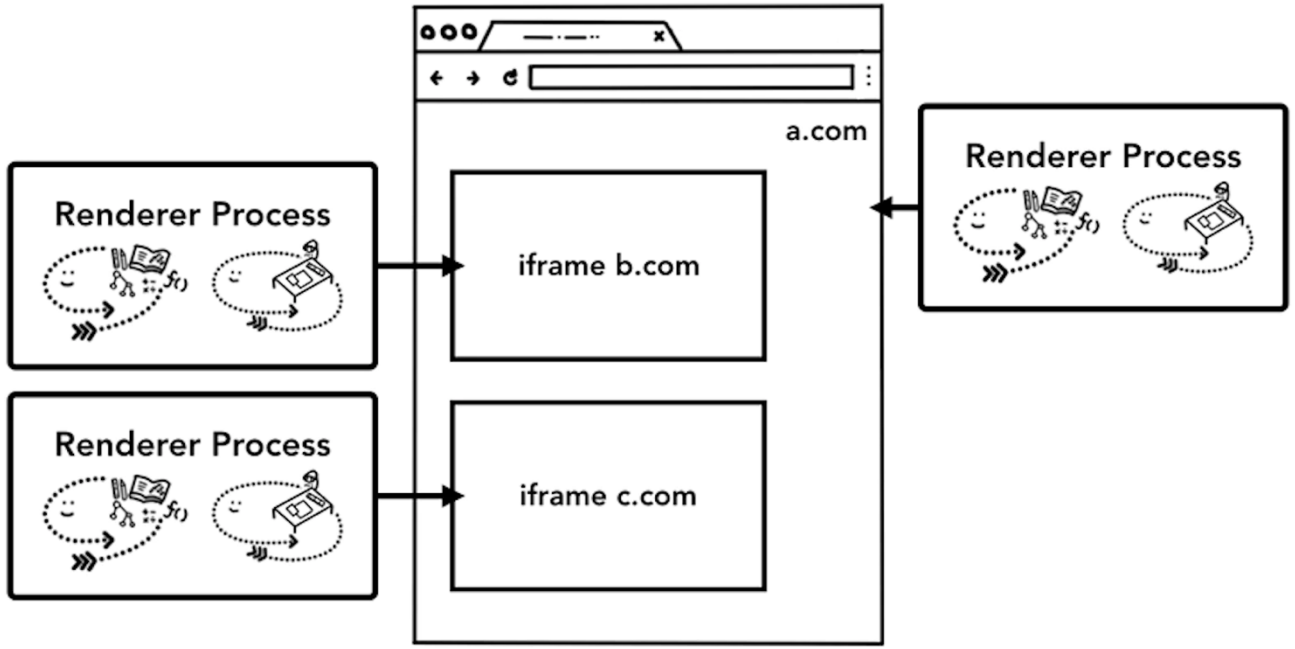
Key Question: *How can our program create and interact with other programs? How does the operating system manage processes?*

- We can use **fork/waitpid/execvp/pipe** to spawn, coordinate, and run other programs in another process.
- The OS has a Process Control Block for each process and could run processes in any order!

Multiprocessing

Why does this matter?

- First look at concurrency challenges – we don't know the order in which processes will execute, and this can impact our code.
- Shows us how shells work!

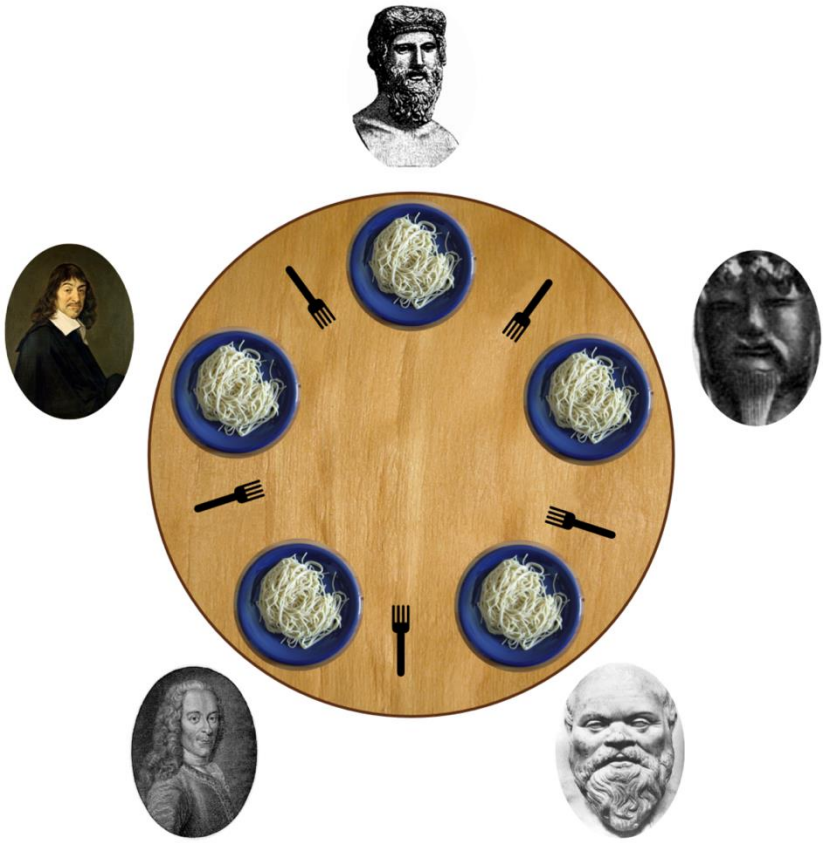


Chrome Site Isolation [source]

Multithreading

Key Question: *How can we have concurrency within a single process? How does the operating system support this?*

- We can spawn threads to have concurrency in a single process, but this presents many challenges with synchronization, race conditions and deadlock.
- The OS tracks and schedules threads (not processes) to run, and switches between them periodically.
- There are various designs for deciding which thread to run next (round robin, SRPT, etc.)



https://commons.wikimedia.org/wiki/File:An_illustration_of_the_dining_philosophers_problem.png

Multithreading

Why does this matter?

- Multithreading can allow us to maximally take advantage of hardware (multicore processors)
- Multithreading has many applications in modern software (e.g. app background download thread, parallelizing spreadsheet recalculation, web server parallelization)
- Helps us understand the behavior of our computers – how even single-core machines can appear to multitask
- Deeper dive into concurrency challenges both in user programs and the OS – synchronization is *hard!* Use only where necessary, and techniques like monitor pattern can help.

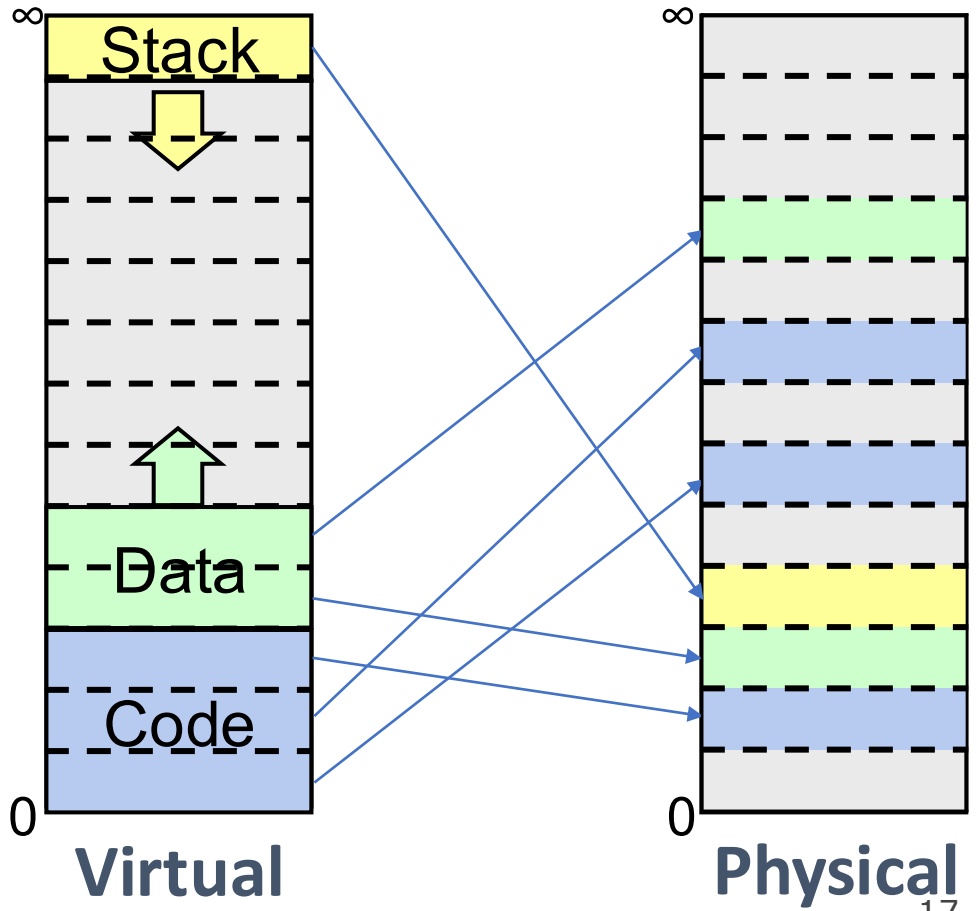
Managing Concurrency

- Processes and threads
- Creating, dispatching
- Synchronization: races, inconsistency, locks, condition variables, monitors, implementations of locks / condition variables
- Scheduling
- Interrupts
- Deadlock

Virtual Memory

Key Question: *How can one set of memory be shared among several processes? How can the operating system manage access to a limited amount of system memory?*

- Virtual memory gives each process its own isolated virtual address space, and the OS maps what's needed to real physical memory.
- The OS can manage physical memory however it wants, including swapping pages to disk if it needs to.



Virtual Memory

Why does this matter?

- Powerful example of **virtualization** – virtual memory creates a world (virtual address space) that doesn't really exist!
- We write our own programs assuming tons of contiguous memory – now we know how this is really enabled.
- Helps us understand the behavior of our computers – thrashing, swap, etc.

Modern Technologies and OSes

Key Question: *How do hardware advances impact the design of operating systems?*

We saw two examples:

- Multicore impacts on scheduling and locks
- Flash storage impacts on filesystems

Why does this matter?

- See examples of how OSes sit at hardware-software boundary – hardware changes can change OSes
- We can more fully understand the impacts of modern technologies on our devices

Ethics and Trust

Key Question: *Who/what do we trust, how do we decide, what do we do when that trust is not upheld, how can we factor trust into what we build, and how does technology impact trust?*

Why does this matter?

- A vital ethical lens through which we can examine large systems such as OSes; immense scale and impact, relied on by many, hard to maintain and debug.
- We must trust *some* things/people/organizations; important to reflect on what we trust and what we value in choosing who/what to trust.
- Chance to reflect on what to do when trust is violated, and how we can incorporate considerations of trust into what we build (e.g., trust by substitution).
- Technology can be involved in agential gullibility, increasing, or decreasing trust

Plan For Today

- **Recap**: Where We've Been
- **Big Ideas**
- What's Next?
- Q&A

Big Ideas

- **Interplay between technology and OSes:** OS at hardware-software boundary
- **Designing with tradeoffs**: not always one "best" way – evaluating pros/cons + priorities
- **Virtualization:** make one thing look like something else, or many of them
- **Managing concurrency:** synchronization is hard!
- **Locality:** the past often predicts the future (scheduling, paging, block cache, etc.)
- **Atomic operations**: take a collection of operations and make them appear as a single indivisible operation (synchronization, file system consistency)
- **Layering:** building higher-level abstractions to hide details (e.g. monitors, file system layers, file descriptors, etc.)
- **System builders wrangle complexity** – solve complex problems with simple interfaces that others can build on (e.g. virtual memory, filesystems)
- **Trust** – we must trust some things/people, evaluate what to trust and how systems can incorporate trust and impact trust

You’re the OS – the game

https://plbrault.com/blog-posts/i-created-the-nerdierst-game-ever/

Plan For Today

- **Recap**: Where We've Been
- Big Ideas
- **What's Next?**
- Q&A

After CS111, you are prepared to take a variety of classes in various areas. What are some options?

Next Steps in Similar Areas

- **CS143** (compilers) – how is a compiler implemented?
  - Model program as a tree
  - Go from code -> assembly
- **CS144** (networking) – how can applications communicate over a network?
  - How can we transmit data across an unreliable network?
  - How does data get to its intended destination?
- **CS145** (databases) – design and use of databases (including transactions, logging)
- **CS149** (parallel computing) – further explore concurrency / challenges in writing parallel software
- **CS155** (security) – how can we find/fix vulnerabilities and improve security?

CS112

Key question: how is an operating system implemented?

- Write parts of a real operating system called *Pintos* – real code in the OS!
- 4 significant projects, done in groups:
  1. Threads
  2. User programs (e.g. system calls)
  3. Virtual memory
  4. Filesystems
- Projects are challenging, but very rewarding
- CS112 is just the "assignments" part of CS140/CS212

Additional option: CS140E – more hardware-oriented course instead of CS112 (140E is to 112 as 107E is to 107)

Other Courses

- CS240 (requires CS112) – Advanced Topics in OSes (old/new papers on OS developments)
- CS244B (requires CS144) – Distributed Systems
- CS190 (requires CS112) – Software Design

Other Courses

- **CS152**: Trust and Safety Engineering
- **CS181:** Computers, Ethics, and Public Policy
- **CS182:** Ethics, Public Policy, and Technological Change
- **CS221:** Artificial Intelligence
- **CS155**: Security
- **EE108**: Digital Systems Design
- **EE180:** Digital Systems Architecture

Plan For Today

- **Recap**: Where We've Been
- Big Ideas
- What's Next?
- **Q&A**

Thank you!

Course Evaluations

We hope you can take the time to fill out the end-quarter course evaluation (open through Mon 3/23). We sincerely appreciate any feedback you have about the course and read every piece of feedback we receive. We are always looking for ways to improve! http://course-evaluations.stanford.edu/

Thank you!

Q&A: What questions do you have?

Respond on PollEV

Respond on PollEv:
pollev.com/cs111