Final Exam

• Next Thur (12/9) 3pm-5pm
• Similar style as midterm, but more time (and probably fewer questions)
• Only covers things from (including) lec8 to lec16
• Based upon lecture material, homeworks, and project
• Doesn’t include anything that we moved to backup slides
• Open book and open everything, but no discussion!
Online Exam

- Canvas quiz
- During the exam
  - Use Piazza to ask private questions

- True of False
- Multiple choices
- Short-answer questions
- Big questions
- We’ll have question(s) related to project2

*Obligatory: Please, do not cheat*
[lec8] CPU Scheduling

- What is non-preemptive and preemptive CPU scheduling? What enables preemptive scheduling?
- What are the goals of CPU scheduling?
- Why is CPU scheduling hard?
Goals (Performance metrics)

- **Minimize turnaround time**
  - avg time to complete a job
  - $T_{\text{turnaround}} = T_{\text{completion}} - T_{\text{arrival}}$

- **Maximize throughput**
  - operations (jobs) per second
  - Minimize overhead of context switches: large quanta
  - Efficient utilization (CPU, memory, disk etc)

- **Short response time**
  - $T_{\text{response}} = T_{\text{firstrun}} - T_{\text{arrival}}$
  - type on a keyboard
  - Small quanta

- **Fairness**
  - fair, no starvation, no deadlock
Scheduling policies

- FIFO
- Round Robin
- SJCF
- SRTCF
- Priority-base
  - Multi-queue
  - Multi-level feedback queue
Practice Problem

- Job 1 – arrives at 0, run for 10
- Job 2 – arrives at 5, run for 20
- Job 3 – arrives at 7, run for 5

- Assume FIFO
  - What’s the average response time?
  - What’s the average turnaround time?
Memory Management

• Why is memory management useful?
  ♦ Why do we have virtual memory if it is so complex?

• What are the mechanisms for implementing MM?
  ♦ Physical and virtual addressing
  ♦ Segmentation and paging
  ♦ Page tables, TLB
  ♦ Swapping

• What are the policies related to MM?
  ♦ Page replacement
main.c -> main.o
math.c -> math.o

Load a.out to mem
Manage mem for proc

Instruction execution
Execute inst w/ virt mem
Translate and access phys mem

Set up and manage virt->phys mem mapping

memory management

Virt Mem

arch

compiler
linker
loader
Virtualizing Memory

- Virtual memory
  - Each process has its own virtual address space
  - Hardware translates virtual address into physical addresses with OS support

- Evolution of techniques
  - Single, fixed physical segment per process (no virt mem)
  - Single segment per process, static relocation (no virt mem)
  - Base-and-bound – dynamic relocating whole process
  - Segmentation – multiple (variable-size) segments with dynamic relocation
  - Paging – small, fixed size pages

- What is internal fragmentation?
- What is external fragmentation?
What are the advantages/disadvantages of paging (over segmentation)?

Know these terms and how to get them
- Virtual page number (VPN), physical page number (PPN)/page frame number (PFN), offset

What are page tables? page table entries (PTE)?

Page tables introduce space and performance overhead

What techniques can be used to reduce their overhead?

How do two-level (multi-level) page tables work?

What are the advantages/disadvantages of multi-level paging (over single-level)?
Multiple-level page tables

Diagram showing the structure of multiple-level page tables.

- Page-map L4
  - Page-map L4 table
  - Page-mp entry
- Page-dir-ptr table
  - Page-dir-ptr entry
- Page-directory table
  - Page-dir entry
- Page-table
  - Page-table entry
  - Physical address
  - Physical page frame number
  - Page offset
  - Main memory
[lec11] TLBs

- What problem does the TLB solve?
- How do TLBs work? Why are TLBs effective?
- How are TLBs managed?
  - What happens on a TLB miss?
- What is the difference between a hardware and software managed TLB?

![Diagram of TLB operation]

Virtual address

VPN | offset
---|---

PFN | offset
---|---

Real page table

Hit

Miss
[lec12] Page Faults

- What is a page fault?
- How does swapping (demand paging) work?

- What is the complete sequence of steps to execute a memory instruction (from CPU instruction all the way to paging from disk)?
  - What is done in hardware, what is done in software?
  - What happens at what time?
Page Fault Handling in demand paging

1. MMU (TLB)
2. Page fault
3. Swap out a victim page to disk
4. Update PTE of victim pg, flush TLB
5. Swap in access page from disk
6. Update PTE of access pg
7. Resume faulting intr

Load M

Inst seq

mem ref

Page Table (TLB)

OS VM subsystem

Physical pages

1 2 3 4 5 6 7
Other VM Topics

- How does memory allocation work?
- How does malloc, brk work?
[lec13] Page Replacement

- What is the purpose of the page replacement algorithm?
- What application behavior does page replacement try to exploit?
- When is the page replacement algorithm used?
- Policies
  - FIFO, LRU, Clock, Enhanced Clock
Page replacement algorithms

- FIFO – replace page loaded furthest in past
- LRU – replace page referenced furthest in past
  » Approximate using PTE reference bit
- Clock – replace page that is “old enough”
- Enhanced Clock – pick clean pages first (for lower miss latency)
[lec14] Storage Devices

- Disk interface
  - How does the OS make requests to the disk?
- Disk performance
  - What are seek, rotation, transfer? What’s the slowest?
- SSD and NVM
  - Performance characteristics? Interface?
File Systems

• Topics
  ♦ Files
  ♦ Directories
  ♦ Disk layout
  ♦ Implementation
  ♦ Buffer Cache
  ♦ Reliability

• What is a file system?
• Why are file systems useful (why do we have them)?
Files System Interface and Operations

- What’s the interface between user and file systems? What’s the interface between file systems and storage devices?
- What is a hierarchical file system?
- What is a directory?
  - How is directory implemented in UNIX?
  - How to do a path walk?
Why do we care about file system layouts? What are the design considerations?

What are the general strategies?
- Contiguous, linked, indexed
- What are the tradeoffs for those strategies?

How does multi-level index work? How is it different from multi-level page table?

What are the ways to keep track of free disk blocks?
Indirect blocks addressing ranges

- Assume block size is 1K
  - a block contains 1024 / 4 = 256 block addresses
- Assume 10 direct block addresses,
  - 1 indirect, 1 double indirect, 1 triple indirect
- 10 direct block address: 10 * 1K = 10K
- 1 indirect block addresses: 256 * 1K = 256K
- 1 double indirect block addresses: 256 * 256K = 64M
- 1 triple indirect block addresses: 256 * 64M = 16G

What happens in accessing block 23, 5, 340?
What’s the maximum size of a file?
[lec16] File System Implementation

• How do we manage information on disk?
  ♦ What are advantages of using disk blocks?
  ♦ What kind of fragmentation does it have?

• What is an inode?
  ♦ What is included in an inode?
  ♦ How are inodes different from directories?

• What are superblock, inode/data bitmap blocks?
- Boot block: contains info to boot OS
- Superblock defines a file system
- An inode for each file => Inode Table
- Data structures to represent free space on disk for both inode and data blocks
  - Bit map: 1 bit per block (sector)
    - How much space does a bit map need for a 4GB disk?
  - A data bitmap and an inode bitmap
How does early-day UNIX file system place inodes and data blocks? What are the drawbacks?

What is a cylinder group? Why cylinder groups?

Where to place inodes, indirect blocks, data blocks, bitmaps? Files under the same directory?
[lec16] File Buffer Cache

- What is the file buffer cache, and why do operating systems use one?
- What is the difference between caching reads and caching writes?
- Write through vs. write back?
- What problems does caching in memory bring?
File System Reliability

- Why do we care about reliability?
- What are different levels of measures we can take to provide reliability?
- What does crash consistency mean?
- What bad things could happen when system crashes during a file operation?
- What are undo and redo logs?
- How does journaling work?
  - Writing to journal, checkpointing, recovery
- What are the ext3 journaling modes? How do they work? What are the pros and cons?
Summary

• Any remaining questions?
The End

• Let’s thank our amazing TAs and Tutors!
  ♦ Jefferson Chien
  ♦ Max Gao
  ♦ Ruohan Hu
  ♦ Yizhou Shan
  ♦ Muyuan Chi
  ♦ Grant Jiang
  ♦ Patrick Lin
  ♦ Joshua Narezo
  ♦ Kyle Wang
  ♦ Fengyuan Wu

• Congratulations on finishing CSE 120!
  ♦ It’s a challenging course, but I hope you found it worthwhile
  ♦ … and that you now look at OSes in a completely new way

• Good luck, and thanks for a great class!