CSE 120
Principles of Operating Systems
Spring 2020

Final Review
Yiying Zhang
Final Exam

- Next Thur (6/11) 7pm-10pm
- Similar style as midterm, but more problems!
- Covers and only covers everything after midterm
- Based upon lecture material, homeworks, and project
- Open book and open everything, but no discussion!

- True of False
- Multiple choices
- Short-answer questions
- Big questions
Online Exam

• Canvas quiz (version A and B)
  ♦ Each version will select randomized questions

• No need for camera this time
  ♦ Sign consent form

• During the exam
  ♦ Use Piazza to ask private questions

• Grading
  ♦ Automatic Canvas grading of T/F and multiple choices
  ♦ Manual grading for the rest

• **Obligatory: Please, do not cheat**
Preparing for Finals

• Go over lectures
  ♦ For better understanding, also read corresponding textbook chapters

• Go over homeworks and practice questions
  ♦ Discussion section yesterday: practice questions
  ♦ Lab hour next Monday: homework 4
  ♦ Discussion section last week (5/27): homework 3

• Do projects on your own
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
  ♦ Partitioning, paging, and segmentation
  ♦ Page tables, TLB
  ♦ Swapping, page buffering
  ♦ Memory sharing, COW

• What are the policies related to MM?
  ♦ Page replacement, thrashing, working sets

• What are the overheads related to providing memory management?
The Big Picture

main.c -> main.o
math.c -> math.o

compiler
linker

a.out

Virt Mem
loader

Load a.out to mem
Manage mem for proc

memory management
Set up and manage virt->phys mem mapping

Instruction execution

arch

Execute inst w/ virt mem
Translate and access phys mem
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
<table>
<thead>
<tr>
<th>Scheme</th>
<th>How</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple uniprogramming</td>
<td>1 segment loaded to starting address 0</td>
<td>Simple</td>
<td>1 process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 segment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No protection</td>
</tr>
<tr>
<td>Simple multiprogramming</td>
<td>1 segment relocated at loading time</td>
<td>Simple, Multiple processes</td>
<td>1 segment/process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No protection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>External frag.</td>
</tr>
<tr>
<td>Base &amp; Bound</td>
<td>Dynamic mem relocation at runtime</td>
<td>Simple hardware, Multiple processes</td>
<td>1 segment/process, External frag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection</td>
<td></td>
</tr>
<tr>
<td>Multiple segments</td>
<td>Dynamic mem relocation at runtime</td>
<td>Sharing, Protection, multi segs/process</td>
<td>More hardware, External frag.</td>
</tr>
</tbody>
</table>
Virtualizing Memory

• What is the difference between a physical and virtual address?
• What is the difference between fixed and variable partitioning?
  ♦ How do base and limit registers work?
• What is internal fragmentation?
• What is external fragmentation?
• What is a protection fault?
Paging

• 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)?
[lec10] Multiple-level page tables
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 fault?
- What is the difference between a hardware and software managed TLB?
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?
[lec11] Page Fault Handling in Demand Paging

- Load M
- mem ref
- Inst seq.
- Page Table (TLB)
- VM subsystem
- physical pages
- fault

Diagram:

- Load M
- mem ref
- Inst seq.
- Page Table (TLB)
- VM subsystem
- physical pages
- fault

Diagram shows the flow of data and control through the memory system, including the load operation, memory reference, instruction sequence, page table, virtual memory subsystem, and physical pages.
Other VM Topics

- What is shared memory?
- What is copy on write?
- How does memory allocation work?
- How does malloc, brk work?
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
  - Belady’s (optimal), FIFO, LRU, Clock, Enhanced Clock
- What is thrashing? What is working set?
[lec12] Summary

• Page replacement algorithms
  ♦ Belady’s – optimal replacement (minimum # of faults)
  ♦ 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)
  ♦ Working Set – keep the set of pages in memory that has minimal fault rate (the “working set”)

• Multiprogramming
  ♦ Global vs. local replacement
  ♦ Thrashing
**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?
What are the two motivations of RAID?

What are the main ideas of RAID?
- Striping, mirroring, parity

RAID levels (0, 1, 4, 5)

How to perform small/big read/write? How many disk accesses?

How to reconstruct a failed disk?
File Systems

• Topics
  ♦ Files
  ♦ Directories
  ♦ Sharing
  ♦ Implementation
  ♦ Buffer Cache

• 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?
• How do soft/hard links work?
• How does create/delete/rename work?
File System Layouts

• 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?
• How 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: $10 \times 1K = 10K$
  - 1 indirect block addresses: $256 \times 1K = 256K$
  - 1 double indirect block addresses: $256 \times 256K = 64M$
  - 1 triple indirect block addresses: $256 \times 64M = 16G$

What happens in accessing block 23, 5, 340?
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 superbloc, inode/data bitmap blocks?
  - What is the master block (superblock)?
• 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?
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?
• How can we use the file buffer cache for read ahead?
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?
Dist Sys, Datacenter, Cloud

• No small or big questions, only T/F and/or multiple choices
Summary

• Any remaining questions?
The End

• Let’s thank our amazing TAs and Tutors!
  ♦ Lihao He
  ♦ Xiao Liu
  ♦ Priyal Rakesh Suneja
  ♦ Jinmou Li

• 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!