CSE 120
Principles of Operating Systems

Spring 2023

Lecture 19: Final Review

Amy Ousterhout

Administrivia

- Project 3
  - Due June 10th – there will be no extensions
- Discussion section this week – go over HW3 and HW4
- Course Evaluation
  - New SET form
  - I really appreciate your feedback
  - Due Saturday June 10th at 8:00 AM

Related Courses

- If you enjoy CSE 120 topics, consider taking other systems courses:

- CSE 123: Computer Networks
- CSE 124: Networked Services
- CSE 125: Software System Design and Implementation
- CSE 127: Computer Security

Today's Outline

- Final logistics
- Final topics
- Practice problems

Final Logistics

- Monday June 12th, FAH 1450 (this room), 3:00-6:00 pm
- You may bring one 8.5”x11” double-sided sheet of notes to the exam
  - Typed or handwritten
- Bring your ID to show to a proctor when you hand in your exam
- Obligatory: complete your exam yourself without assistance from others
  - Sign an agreement on the first page indicating this

Final Content

- Based on lectures, homework, and programming projects in the second half of the quarter
  - Lectures 10-18
  - Also some concepts from lectures 2 and 3
  - Homeworks #3 and #4
  - Projects 2 and 3

Types of Questions

- True or False
- Multiple choice
- Short answer
- Larger problems
  - May include writing code
  - May include drawing diagrams

Today's Outline

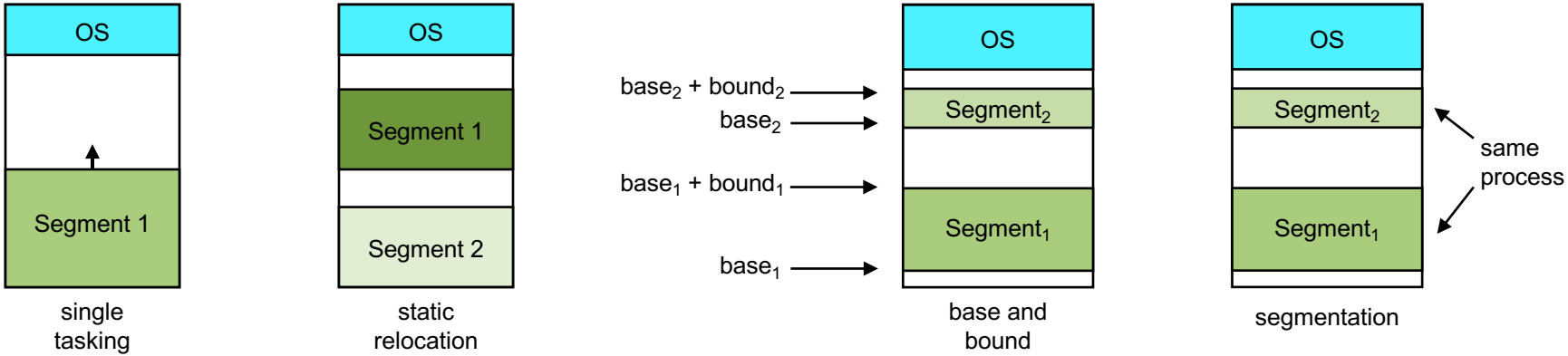
- Final logistics
- Final topics
- Practice problems

Memory Management

- Why is memory management useful?
  - Why do we have virtual memory?
- What are the mechanisms for implementing memory management?
  - Physical and virtual addressing
  - Partitioning, paging, and segmentation
  - Page tables, TLB
- What are the policies related to memory management?
  - Page replacement
- What are the overheads related to providing memory management?

Virtualizing Memory

- What is the difference between a physical and a virtual address?
- What is the difference between different approaches?

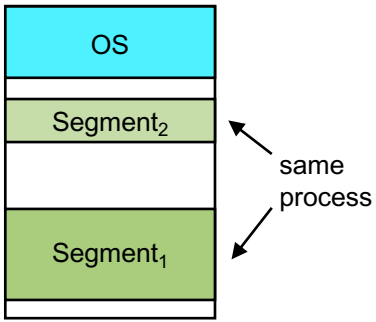


Virtualizing Memory

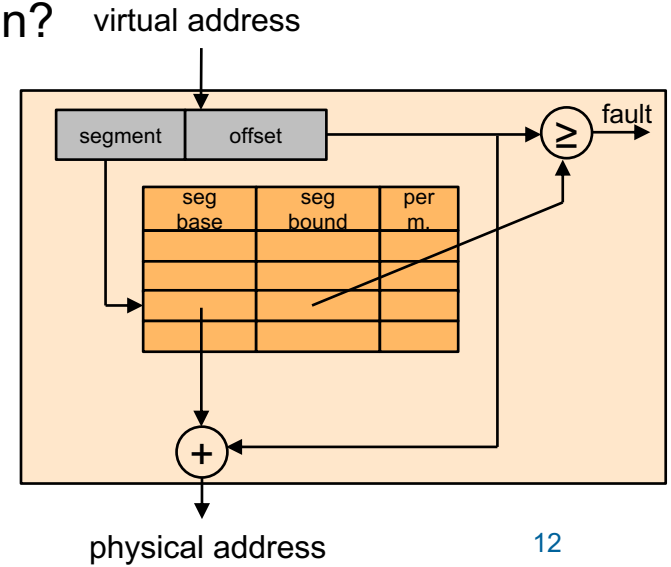
- What is internal fragmentation?
- What is external fragmentation?
- What kinds of faults are related to memory management?

Segmentation

- What is segmentation?
- How does it compare/contrast with paging?
- What are its advantages/disadvantages with respect to paging?
- How do you translate addresses with segmentation?
  - What is a segment table?

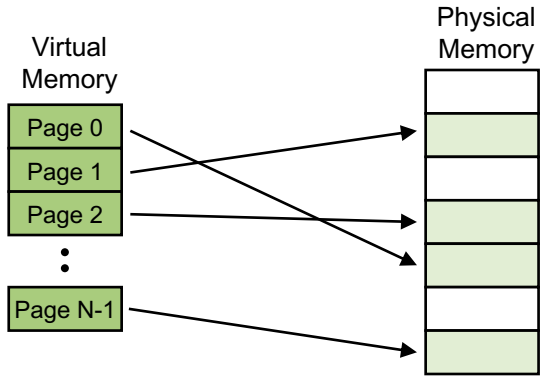


segmentation



Paging

- What is paging?
- What are the pros/cons of paging?
- What are page tables?
- What are each of the following?
  - Page table entries (PTEs)
  - Virtual page number (VPN)
  - Physical page number (PPN)/page frame number (PFN)
  - Offset
- How do you break down a virtual address into a page number and offset?
- How did you implement paging in Nachos?

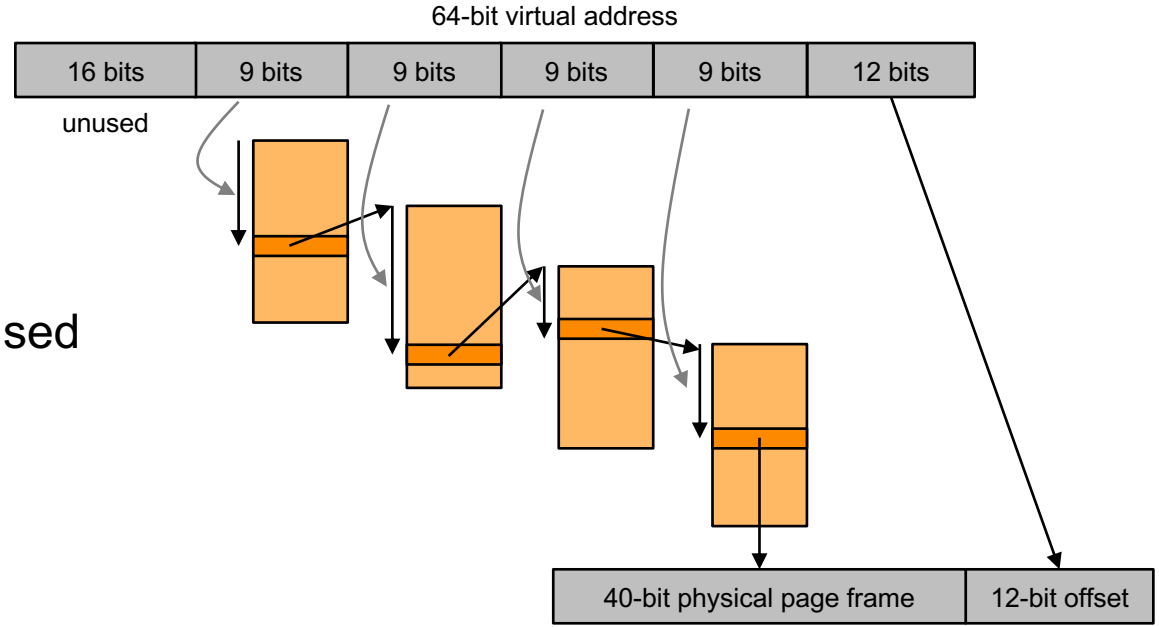


Page Table Entries

- What is a page table entry?
- What are all the PTE bits used for?
  - Modify (dirty)
  - Reference (used)
  - Valid
  - Protection
  - Page frame number (PFN)

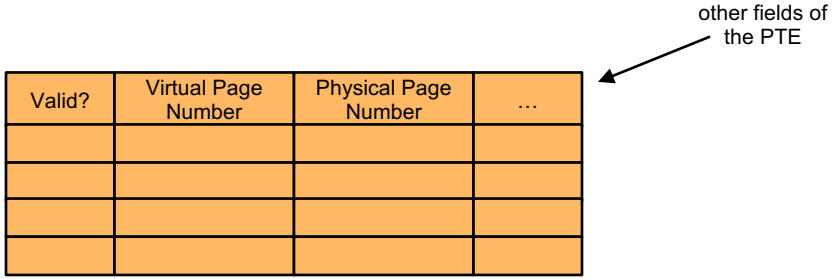
Page Tables

- Page tables introduce overhead
  - Space for storing them
  - Time to use them for translation
- What techniques can be used to reduce their overhead?
- How do multi-level page tables work?



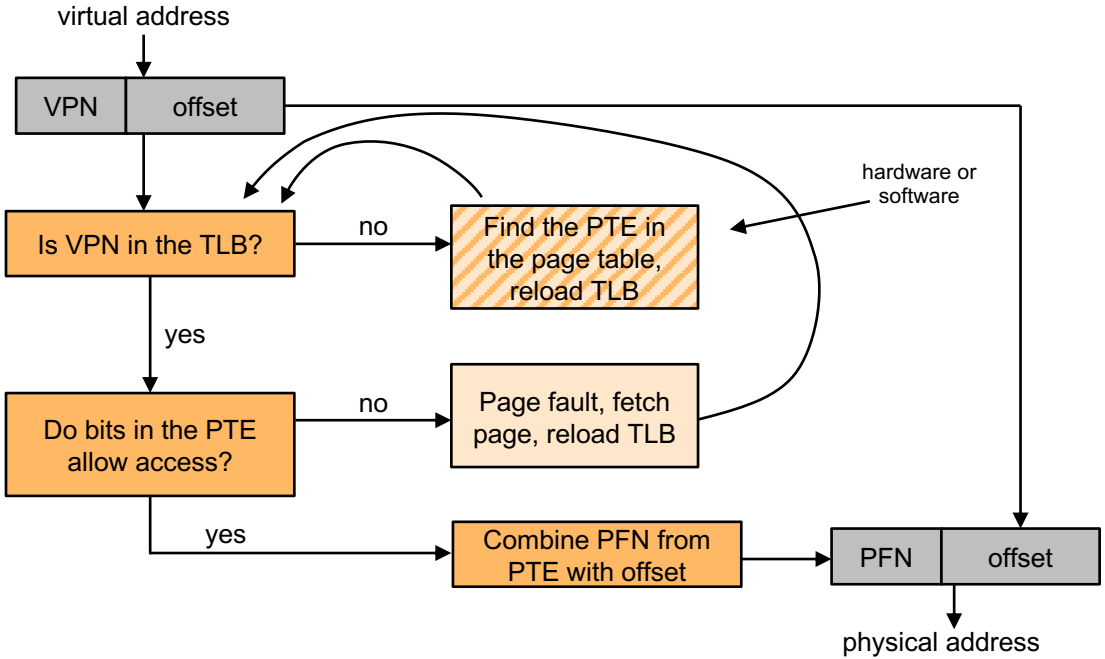
TLBs

- 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?



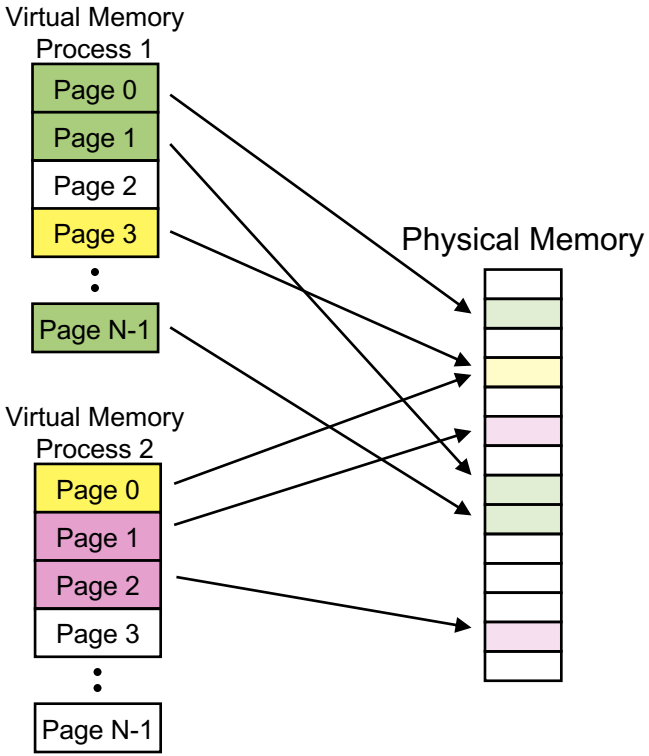
Page Faults

- What is a page fault?
  - How is it used to implement demand-paged virtual memory?
- What is the complete sequence of steps for translating a virtual address to a physical address?
  - What is done in hardware, what is done in software?



Advanced Virtual Memory Topics

- What is shared memory?
- What is copy on write?

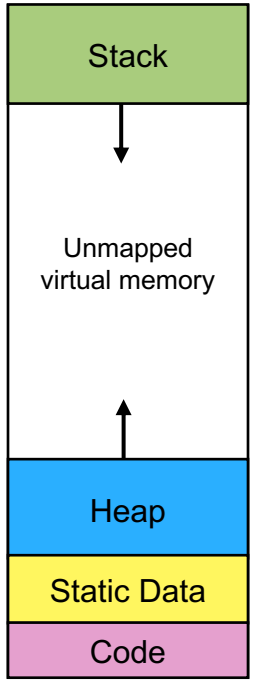


Page Replacement

- 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?
- How do each of the following work?
  - Belady's MIN, FIFO, LRU, Clock
- What is Belady's anomaly? Working set? Thrashing?
- Locality
  - What is spatial locality?
  - What is temporal locality?

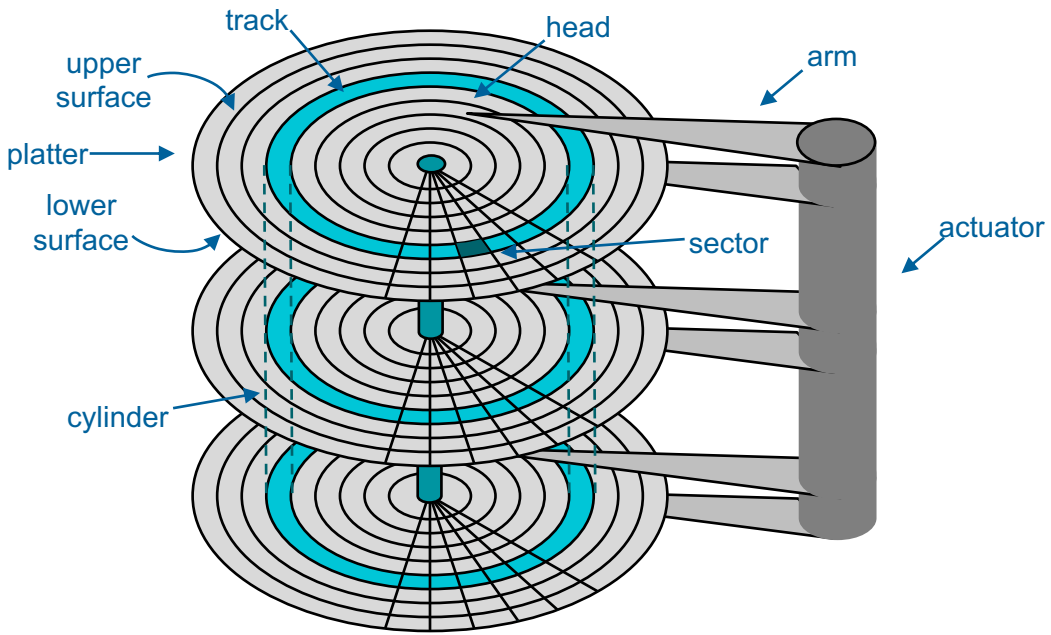
Virtual Memory Allocation

- How do we manage and grow the stack?
- How do we manage and grow the heap?



Disk

- What are sectors? Tracks? Heads?
- What are the components of disk latency?
- How does disk access time compare to memory access time?
- How do disk scheduling policies work?
  - FIFO, shortest seek time first, elevator (SCAN)
- What is the goal of RAID?

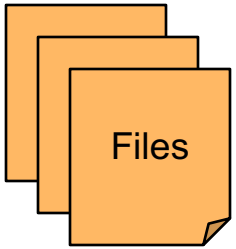


File Systems

- What is a file system?
- Why are files useful (why do we have them)?

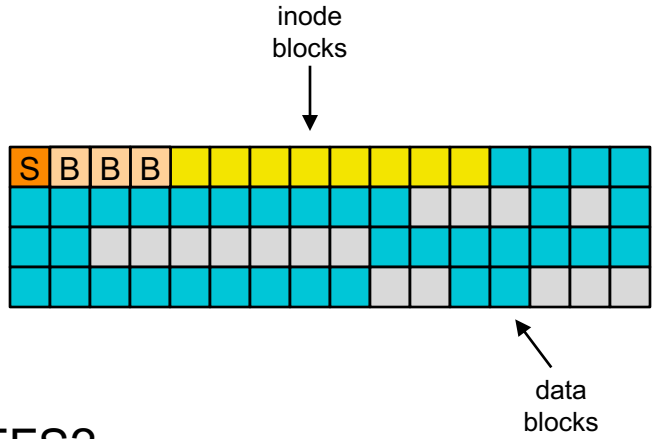
Files and Directories

- What is a file?
  - What operations are supported?
  - What characteristics do they have?
  - What are file access methods?
- What is a directory?
  - What are they used for?
  - What is a directory entry?



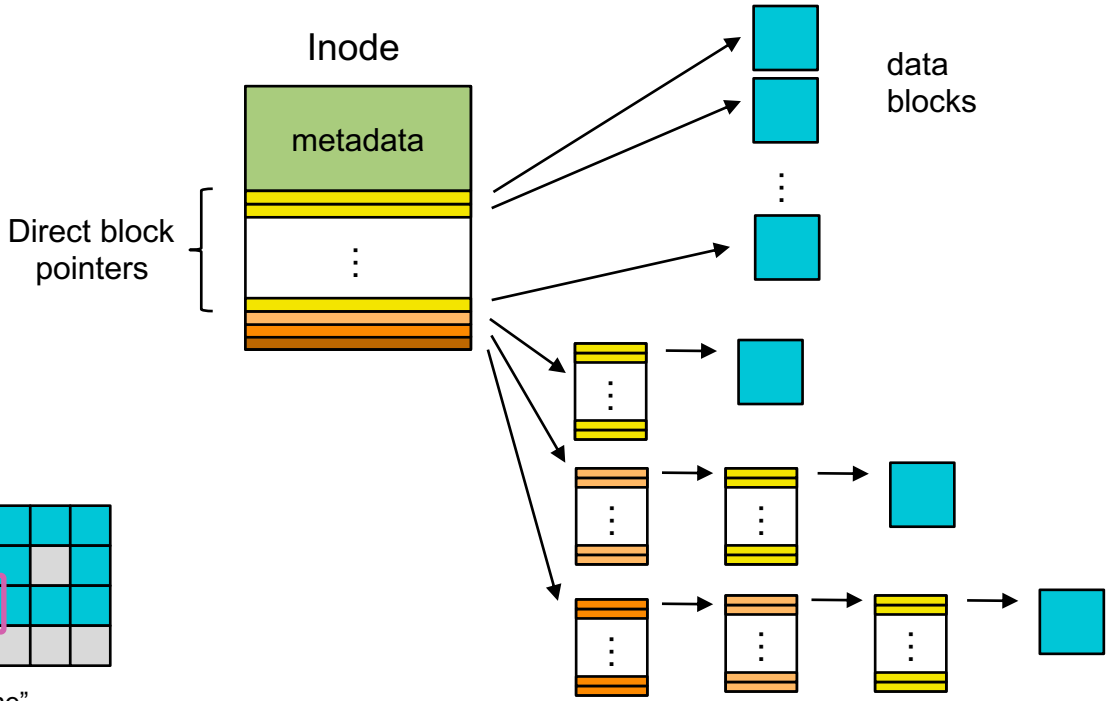
File System Layout

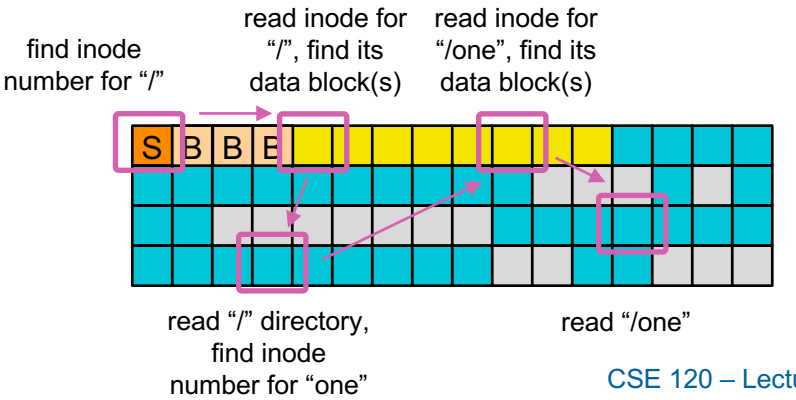
- How do we manage information on disk?
  - What are the advantages of using disk blocks?
  - What kind of fragmentation do they have?
- What are the general strategies?
  - Contiguous, linked, indexed
- What are the tradeoffs of those strategies?
- What are bitmap blocks used for?
- What is the superblock?
- How are cylinder groups/block groups used in FFS?



Inodes and Path Translation

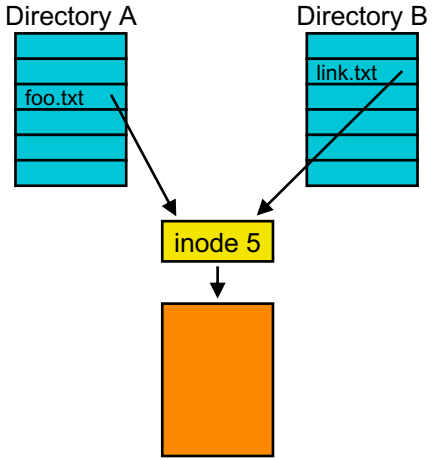
- What is an inode?
  - How are inodes different from directories?
- How are inodes and directories used to do path translation to find files?

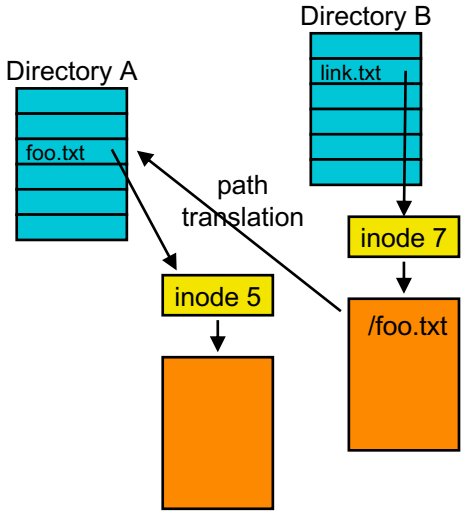




File Operations

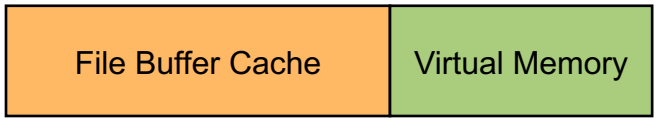
- How do hard links work?
- How do soft links work?
- How do create, delete, and rename work?





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?
- What are the tradeoffs of using memory for a file buffer cache vs. virtual memory?

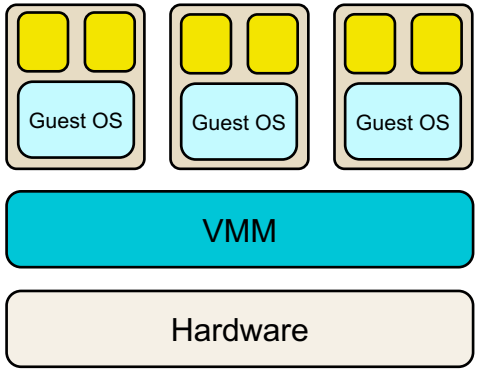


Crash Consistency

- Why is crash consistency a problem?
- What are examples of things that can go wrong with the file system when we crash?
- How can we keep our file system consistent?
  - File system checker (`fsck`)
  - Ordered writes
  - Journaling
- What are the tradeoffs of these different approaches?

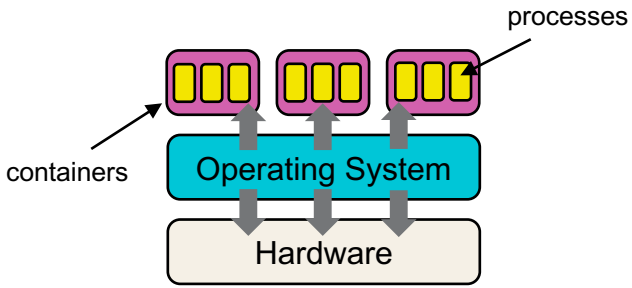
Virtual Machines

- What is a Virtual Machine Monitor (VMM)?
  - What abstraction do they provide to virtual machines (VMs)?
- Why are virtual machines useful?
- What is the difference between type 1 and type 2 hypervisors?
- How does trap-and-emulate work?
- Why is virtualizing x86 challenging and what techniques do we use to overcome this challenge?



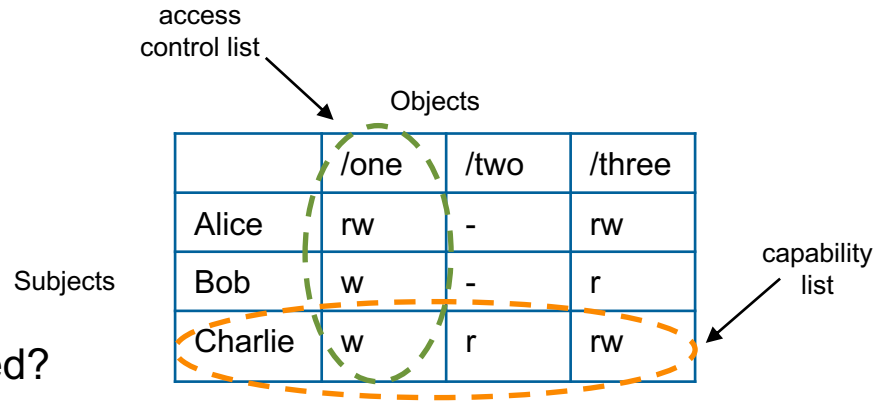
Containers

- How is a container similar to/different from a process or a virtual machine?
- What are namespace isolation and resource management?
  - How do containers use them?



Protection

- What are the principles of protection?
- How is user identity used in protection?
- What are access control lists and capabilities?
  - How do operating systems use each?
  - How are they represented and implemented?
  - What are the pros/cons of each?
- How do we implement protection for files and virtual memory?
  - How do we check whether a process is allowed to use files or memory?



Today's Outline

- Final logistics
- Final topics
- Practice problems

File System Crash Consistency (Q)

- Your goal is to implement a `truncate` system call, which reduces the size of a file with file descriptor `fd` by dropping the last `length` bytes:
  - `truncate(int fd, int length)`
- Which blocks in the file system will be modified by this system call?
- What could go wrong after a crash (without crash-consistency techniques)?
- How could you solve this problem with ordered writes?
- How could you solve this problem with journaling?

File System Crash Consistency (A)

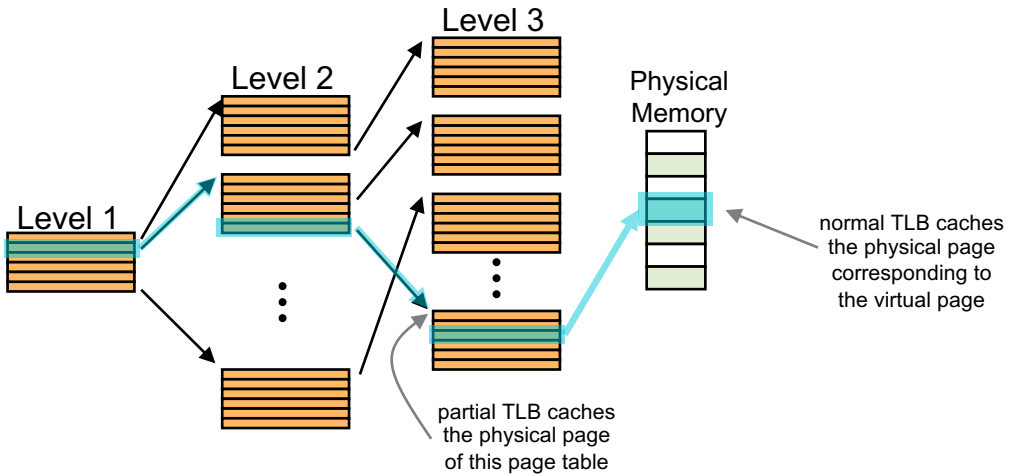
- Which blocks in the file system will be modified by this system call?
  - inode block for fd and bitmap blocks for the data blocks that will be dropped
- What could go wrong after a crash (without crash-consistency techniques)?
  - If the bitmap is updated first, another file could allocate the same data blocks
  - If the inode is updated first, the bitmap will indicate that the block(s) are in use even though they are not (space leak)
- How could you solve these problems with ordered writes?
  - Write the inode block first and then the bitmap blocks (and run `fsck` periodically)
- How could you solve this problem with journaling?
  - Write one transaction to the journal that contains the inode block and bitmap blocks

TLBs and Multi-level Page Tables (Q)

- Your friend proposes a new hardware structure that they call a partial TLB. A partial TLB maps virtual pages to the physical page of their final page map (bypassing intermediate levels of page tables). It has the same number of entries as a regular TLB.
- Describe a workload where this would perform worse than a regular TLB.
- Describe a workload where this would perform better than a regular TLB.
- Do you think partial TLBs are a good approach overall? Why/why not?

TLBs and Multi-level Page Tables (A)

- A normal TLB maps from a virtual page number to a physical page number
- The proposed partial TLB maps from a virtual page number to the physical page number of the last-level page table
  - Address translation still requires one memory access even when you hit in the partial TLB



TLBs and Multi-level Page Tables (A)

- Describe a workload where this would perform worse than a regular TLB.
  - A workload where the number of pages in the working set is ≤ the number of entries in a regular TLB
- Describe a workload where this would perform better than a regular TLB.
  - A workload where the number of pages in the working set is > the number of entries in a regular TLB but the number of last-level page tables is ≤ the number of entries in a partial TLB (some pages use the same page table)
- Do you think partial TLBs are a good approach overall? Why/why not?
  - No, TLBs already have high hit rates and this approach would always add an extra memory access to translation

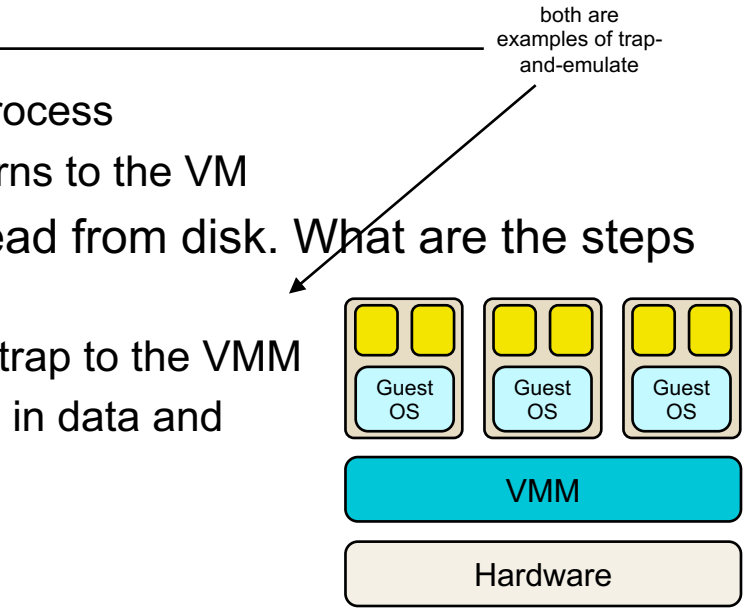
Virtual Machines (Q)

- For these questions, assume a type 1 hypervisor, no binary translation, no paravirtualization, and no hardware support for virtualization.
- Consider a process running in a VM that tries to issue the `halt` instruction. What are the steps for handling this?

- Consider a Guest OS in a VM that tries to read from disk. What are the steps for handling this?

Virtual Machines (A)

- Consider a process running in a VM that tries to issue the `halt` instruction. What are the steps for handling this?
  - Halt instruction causes a trap to the VMM
  - VMM calls into the Guest OS which kills the process
  - Guest OS returns to the VMM which then returns to the VM
- Consider a Guest OS in a VM that tries to read from disk. What are the steps for handling this?
  - Attempting to access disk hardware causes a trap to the VMM
  - VMM emulates the behavior of the disk (reads in data and puts it where the guest OS expects it to be)
  - VMM returns to the Guest OS in the VM



Segmentation and Paging (Q)

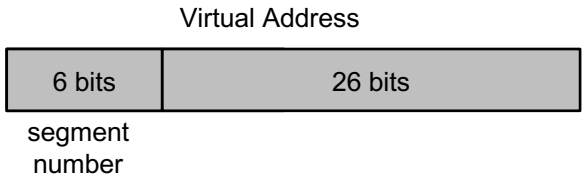
- Consider a virtual memory system that uses both segmentation and paging. The virtual address space is first divided into segments and then further divided into pages.
  - Virtual addresses are 32 bits and a page is 1024 (2^{10}) bytes.
  - A page table must fit on one physical page, the size of a PTE is 4 bytes, and it has 12 bits of flags.
  - One process may have at most 64 (2^{6}) segments.
- What is the maximum size of a virtual memory segment?
- How many pages are there per segment?
- How many levels of page tables are required for virtual address translation?
- Draw an address translation diagram showing the tables used and how the bits of the virtual and physical addresses are divided.

Segmentation and Paging – Intuition, part 1

- We want to be able to use the entire address space (2^{32} bytes)
- One way to approach this problem is to think about the sizes of different components.
  - We are given that the address space is 2^{32} bytes, there are 2^{6} segments, and a page is 2^{10} bytes
- Another way to approach this problem is to think about the bits in the virtual address
  - From above we can see that there are 32 bits in the virtual address, 6 bits in the segment number, and 10 bits in the page offset
- We can use either of these approaches to answer the first 2 questions

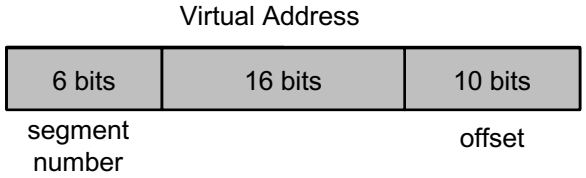
Segmentation and Paging (A)

- Consider a virtual memory system that uses both segmentation and paging. The virtual address space is first divided into segments and then further divided into pages.
  - Virtual addresses are 32 bits and a page is 1024 (2^{10}) bytes.
  - A page table must fit on one physical page, the size of a PTE is 4 bytes and it has 12 bits of flags.
  - One process may have at most 64 (2^6) segments.
- What is the maximum size of a virtual memory segment?
  - 2^6 segments so 6 bits for segment number
  - Leaves 32 – 6 = 26 bits for addressing within a segment
  - Segment size is 2^{26} bytes



Segmentation and Paging (A)

- Consider a virtual memory system that uses both segmentation and paging. The virtual address space is first divided into segments and then further divided into pages.
  - Virtual addresses are 32 bits and a page is 1024 (2^{10}) bytes.
  - A page table must fit on one physical page, the size of a PTE is 4 bytes and it has 12 bits of flags.
  - One process may have at most 64 (2^{6}) segments.
- How many pages are there per segment?
  - Segment size is 2^{26} bytes (previous slide)
  - Page size = 2^{10} bytes
  - 2^{26} bytes per segment / 2^{10} bytes per page = 2^{16} pages per segment

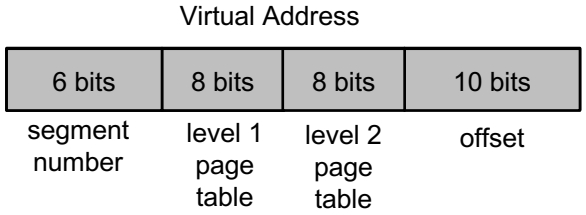


Segmentation and Paging – Intuition, part 2

- Each page table can only fit so many entries in it
- If one page table per segment does not have enough entries for all of the pages in a segment, then we will need multiple levels of page tables
- Think about:
  - How many entries can fit in one page table?
  - When we add an additional level of page tables, how does that change the number of pages that we can address?
    - » E.g., if page tables can fit 4 entries, with one level we can address 4 pages. With 2 levels we can address 4x4=16 pages. With 3 levels we can address 4x4x4=64 pages…

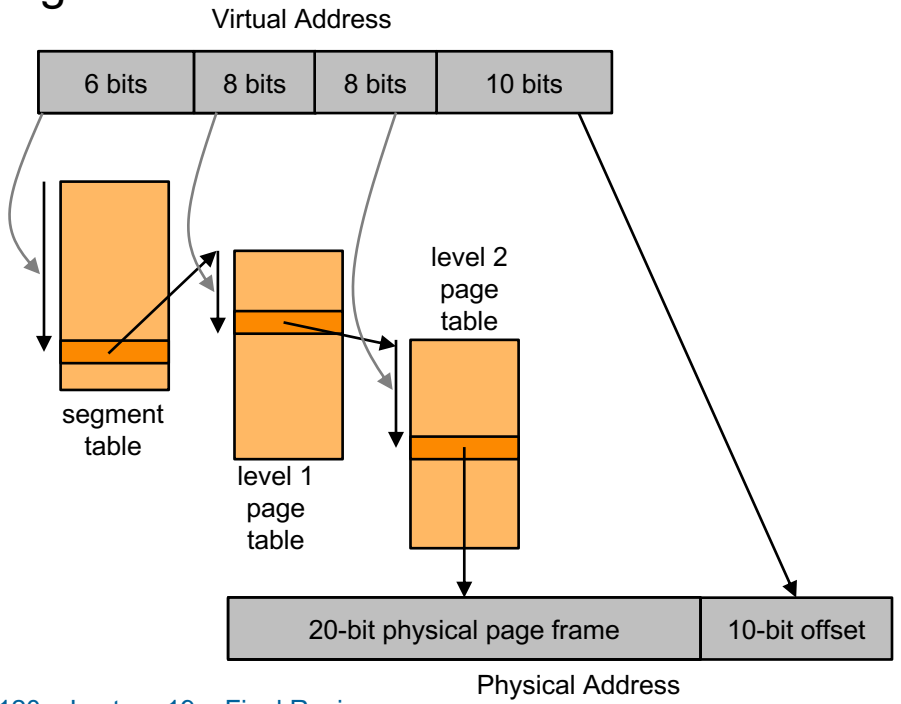
Segmentation and Paging (A)

- Consider a virtual memory system that uses both segmentation and paging. The virtual address space is first divided into segments and then further divided into pages.
  - Virtual addresses are 32 bits and a page is 1024 (2^{10}) bytes.
  - A page table must fit on one physical page, the size of a PTE is 4 bytes and it has 12 bits of flags.
  - One process may have at most 64 (2^6) segments.
- How many levels of page tables are required for virtual address translation?
  - Page table size = 2^{10} bytes and 4 bytes per PTE
  - 2^8 PTEs per page
  - 16 bits for addressing page tables / 8 bits per level
  - 2 levels of page tables



Segmentation and Paging (A)

- Draw an address translation diagram.
  - How many bits per PFN?
  - 32 bits per PTE
  - 12 bits of flags per PTE
  - 32 – 12 = 20 bits for PFN



Page Replacement (Q)

- Consider a machine with 8 GB of memory. The OS's replacement policy is: when a page needs to be evicted, evict the page that has been in memory the longest. The owner notices that some workloads incur a lot of page faults and installs 2 GB more memory. Answer true/false for the following:
- There are workloads for which the extra memory will *decrease* the number of page faults
- There are workloads for which the extra memory will *have no effect on* the number of page faults
- There are workloads for which the extra memory will *increase* the number of page faults

Page Replacement (A)

- There are workloads for which the extra memory will *decrease* the number of page faults
  - True – the working set might fit into 10 GB but not 8 GB
- There are workloads for which the extra memory will *have no effect on* the number of page faults
  - True – the workload might loop through a large amount of memory (e.g., 12 GB) such that every access incurs a page fault
- There are workloads for which the extra memory will *increase* the number of page faults
  - True – this is Belady's anomaly, where for some access patterns and replacement policies (including FIFO), increasing the size of a cache can increase the number of cache misses

Protection (Q & A)

- Explain what a capability is and give an example, either from operating systems or from real life.
  - A capability specifies what actions a subject is allowed to perform on an object.
  - Examples: file descriptor, PTE, physical key
- Explain what an access control list is and give an example, either from operating systems or from real life.
  - An access control list specifies which users are allowed to perform what actions on a given object
  - Examples: UNIX's permissions on files, list of people allowed to enter an event

Any other questions?