

---

# CSC369

# Operating Systems

Spring 2007

Final Review

# Overview

- Final mechanics
- Processes & Threads
- Concurrency, Synchronization & Deadlock
- Scheduling
- Memory management
  - Paging
  - Page replacement
- Disk I/O
- File systems (including distributed)
- Security

# Final Mechanics

- Bulk of the final covers material after midterm
  - Memory management, file systems, storage mgmt, distributed file systems, security
- Some material on concurrency, synchronization
  - Synch primitives, synch problems
- Based upon lecture material, assigned readings & projects
- **Closed book, one 8.5"x11" double-sided sheet of notes**
  - Just one!
  - You **MUST** prepare this sheet manually. No digital reproductions.

# Operating Systems (general)

- Why do we have operating systems? What is their purpose?
- What tradeoffs do operating system designers need to make?
- What design principles guide the development of an operating system?

# Processes & Threads

- What is a process? What is a thread?
- What is the difference between user-level threads and kernel-level threads?
  - When is one type preferable to the other?
- How are new processes created?  
Controlled? Deleted?

# Concurrency

- Synchronization primitives
  - Software solutions
  - Hardware instructions
  - Semaphores
  - Locks
  - Monitors
- Deadlocks
  - Conditions for deadlock
  - Prevention/avoidance/detection & recovery

# Scheduling

- Goals in developing a “good” scheduling algorithm
- Know the properties of different algorithms we discussed

# 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
- What are the policies related to MM?
  - Page replacement
- What are the overheads related to providing memory management?

# 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

- How is paging different from partitioning?
- What are the advantages/disadvantages of paging?
- What are page tables?
- What are page table entries (PTE)?
- Know these terms
  - Virtual page number (VPN), page frame number (PFN), offset
- Know how to break down virtual addresses into page numbers, offset

# Page Table Entries

- What is a page table entry?
- What are all of the PTE bits used for?
  - Dirty (also known as modify)
  - Reference
  - Valid
  - Protection

# Segmentation

- What is segmentation?
- How does it compare/contrast with paging?
- What are its advantages/disadvantages with respect to partitioning, paging?
- What is a segment table?
- How can paging and segmentation be combined?

# 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 two-level (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 fault?
- 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, from a TLB miss to paging in from disk, for translating a virtual address to a physical address?
  - What is done in hardware, what is done in software?

# Advanced Mem Management

- What is shared memory?
- What is copy on write?
- What are memory mapped files?

# 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?
- Understand
  - Belady's (optimal), FIFO, LRU, LRU Clock (second chance), Working Set, Page Fault Frequency
- What is thrashing?

# Disk

- Understand the memory hierarchy concept, locality
- Physical disk structure
  - Platters, surfaces, tracks, sectors, cylinders, arms, heads
- Disk interface
  - How does the OS make requests to the disk?
- Disk performance
  - What steps determine disk request performance?
  - What are seek, rotation, transfer?

# Disk Scheduling

- How can disk scheduling improve performance?
- What are the issues in disk scheduling?
  - Response time, throughput, fairness
- Understand
  - FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK

# File Systems

- Topics
  - Files
  - Directories
  - Sharing
  - Protection
  - Layouts
  - Buffer Cache
- What is a file system?
- Why are file systems 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?
  - How are they implemented?
  - What is a directory entry?
- How are directories used to do path name translation?

# Protection

- What is file protection used for?
- How is it implemented?
- What are access control lists (ACLs)?
- What are capabilities?
- What are the advantages/disadvantages of each?

# File System Layouts

- What are file system layouts used for?
- What are the general strategies?
  - Contiguous, linked, indexed?
- What are the tradeoffs for those strategies?
- How do those strategies reflect file access methods?
- What is an inode?
  - How are inodes different from directories?
  - How are inodes and directories used to do path resolution, find files?

# 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. VM?

# Advanced FS Topics

- What is FFS, and how is it an improvement over the original Unix file system?
- What are the issues in designing a distributed file system?
  - What is location transparency? Location independence?
  - What are the options for caching? (client vs. server, client memory vs. client disk)
  - How is consistency maintained?
  - What new failures/errors are possible?
  - How do NFS and AFS compare with respect to these issues?
- What is RPC, and how is it implemented?

# Security

- Types of security threats
- Types of vulnerabilities
- Defenses
- Malicious software
- Design principles for secure systems

# Conclusion

- Congratulations on surviving CSC 369
  - It's a tough course, but I hope you found it worthwhile
- Check the exam schedule CAREFULLY
- Good luck, and thanks for a great class!

# If OSs were Beer...

Taken from <http://www.topb.com/humor/systems.shtml>

## DOS Beer

- Requires you to use your own can opener, and requires you to read the directions carefully before opening the can. Originally only came in an 8-oz. can, but now comes in a 16-oz. can. However, the can is divided into 8 compartments of 2 oz. each, which have to be accessed separately. Soon to be discontinued, although a lot of people are going to keep drinking it after it's no longer available.

## MAC Beer

- At first, came only a 16-oz. can, but now comes in a 32-oz. can. Considered by many to be a "light" beer. All the cans look identical. When you take one from the fridge, it opens itself. The ingredients list is not on the can. If you call to ask about the ingredients, you are told that "you don't need to know." A notice on the side reminds you to drag your empties to the trashcan.

---

## Windows 2000 Beer

- The manufacturer of the Windows line of beers says this will be "the" beer, if they can just finish playing with the ingredients. This beer will have many ingredients of Windows 95/98 and NT beers. Many drinkers in the future will be forced to drink this when they get thirsty since they won't be able to find Windows 95 or 98 or NT beer on the shelves. According to manufacturer it's combines the greatest taste ever with almost no calories. Only one problem, the cans explode without warning and take out half the refrigerator with them.

## Unix Beer

- Comes in several different brands, in cans ranging from 8 oz. to 64 oz. Drinkers of Unix Beer display fierce brand loyalty, even though they claim that all the different brands taste almost identical. Sometimes the pop-tops break off when you try to open them, so you have to have your own can opener around for those occasions, in which case you either need a complete set of instructions or a friend who has been drinking Unix Beer for several years.

---

## Linux Beer

- LINUX beer tastes just like Unix beer. Like Unix beer, Linux beer is intended for expert beer drinkers only. It originally had no pop tops or cans because you had to brew it yourself. First you would get a recipe and some yeast from a Unix guru. Then you would go plow a field, plant your barley and hops. After harvest you would take your Kernels and put them into a barrel full of water, then you just add your yeast close the lid, and let your beer compile. After all this you have what experts claim to be one of the Worlds Best Beers. Linux beers do not normally explode but many brewers have been known to. Linux beer is now available from some Micro Brewerys in handy pop top versions for easy drinking by beginner Unix or Linux beer drinkers. Keep your can openers handy.