CSE 451: Operating Systems

Section 10

Project 3 wrap-up, final exam review
Final exam review

Disclaimer: This is not guaranteed to be everything that you need to know for the final. This is an overview of major topics we covered in the course.

You are responsible for all the readings and the slides only up to what we covered in class.
Exam Coverage

* Lectures: Modules 1 – 18
  * Everything from the intro to reliable storage is fair game (will not include Networking/RPC/DFS/VMMs)
  * Chapters 1 – 14 in the textbook

* Extra Readings: 2 questions from extra readings

* Projects: 2 questions based on the projects
Major Topics

- Kernels – Micro, Monolithic, etc
- Processes – fork, vfork, execve
- User and Kernel level threads
- Scheduling
- Paging, caching
- Memory Management
More Topics

- Deadlock
- Race conditions and synchronization variables
- File systems
- Projects 1 - 3
Synchronization Variables

- Locks, mutexes, semaphores, condition variables and monitors
  - Mutexes
    - Provide a waiting queue for threads that are waiting on a lock
  - Condition Variables
    - A higher level construct than mutexes. They help manage the waiting of threads by allowing them to wait until a given condition is true
    - Signal and broadcast
  - Monitors
    - Two main different types, Hoare and Mesa monitors.
    - Provides object like abstraction to synchronization. Manages condition variables and locks as well as provides methods for accessing shared memory.
    - Should be familiar with both types: http://en.wikipedia.org/wiki/Monitor_(synchronization)
Thread management

* Queues
  * Why do thread libraries make use of queues?

* Synchronization
  * What are the mechanisms for protecting critical sections, how do they work, and when should one be used over another?

* Preemption
  * What is preemption and how does the process of one thread preempting another work?
Different scheduling techniques:
- First in first out, round robin, shortest processing time first, priority, multi-level feedback queue
- What are the advantages and disadvantages of each
- Starvation and fairness
- Measure of response time

What do most current systems use?
Threads

- Difference between user and kernel level threads
  - Can user level threads run across multiple processors?

- Performance differences between user / kernel level threads

- What are the benefits of using kernel over user level threads, visa-versa
  - Kernel level threads allow for scheduling across multiple processors
  - User level threads are lightweight and run in user space
Kernels

- Different types of OS kernels
  - Micro vs Monolithic
- What are the benefits of each
- What operations need to happen in the kernel vs user space?
  - Interactions with hardware
  - Kernel trap
  - System calls
  - Exceptions
Processes

※ Should know the difference between processes and threads

※ What is the difference between fork and forkv

※ Copy on write?
Memory management

- Purposes:
  - Resource partitioning / sharing
  - Isolation
  - Usability

- Paging

- Segmentation
Virtual memory

What happens on a virtual memory access?
Virtual memory

- What happens on a virtual memory access?
  - Address translation: who performs it?
    - Page table lookup
    - Translation Lookaside Buffer (TLB)
  - Page fault?
    - Page replacement
    - Process/queue management

- How does all of this overhead pay off?
  - **Locality!** Both temporal (in time) and spatial (nearby).
Virtual memory

Note: Each process has its own page table!
Page replacement

- Algorithms:
  - Belady, FIFO, LRU, LRU clock / NRU, random, working set...
  - Local vs. global

- How/why are any of these better or worse than the others?

- What happens when paging goes wrong?
  - Thrashing, 10-year old computers running XP?
Advanced virtual memory

What problem does a TLB address?

What problem do two-level page tables address?

What’s the key concept?
Advanced virtual memory

* What problem does a TLB address?
  * Increases speed of virtual address translation

* What problem do two-level page tables address?
  * What’s the key concept?
    * Indirection
Secondary storage

- Memory forms a **hierarchy**
- Different levels of disk abstraction:
  - Sectors
  - Blocks
  - Files
- What factor most influences the ways that we interact with disks?
Secondary storage

* Memory forms a hierarchy

* Different levels of disk abstraction:
  * Sectors
  * Blocks
  * Files

* What factor most influences the ways that we interact with disks?
  * Latency
Memory hierarchy

Each level acts as a cache of lower levels

(Stats more or less for Core i7 3770)
File systems

What does a file system give you?

- Useful abstraction for secondary storage
- Organization of data
  - Hierarchy of directories and files
- Sharing of data
File system internals

- Directories
- Directory entries
- Inodes

Files:
- One inode per file
- Multiple directory entries (links) per file
Inode-based file system

Sequence of steps when I run `echo "some text" > /home/jay/file.txt`?

- Open file:

- Write to file:

- Close file:
Inode-based file system

Sequence of steps when I run `echo "some text" > /home/jay/file.txt`?

- Open file:
  - Get inode for / -> get data block for /
  - Read directory entry for / -> get inode for /homes
  - Repeat… -> get data block for file.txt, check permissions

- Write to file:
  - Modify data block(s) for file.txt in buffer cache

- Close file:
  - Mark buffer as dirty, release to buffer cache
  - Kernel flushes dirty blocks back to disk at a later time