Architectural Support

• Privileged instructions
  – what are they?
  – how does the CPU know whether to execute them?
  – why do they need to be privileged?
  – what do they manipulate?

• Protected memory
  – what are the various ways it can be implemented?
  – “protected addresses”

• System call
  – what are the steps in handling?

• Interrupts, exceptions, traps
  – definition of each
  – what are the steps in handling each?
OS Structure

• What are the major components of an OS?
• How are they organized?
  – what is the difference between monolithic, layered, microkernel OS’s?
    • advantages and disadvantages?
Memory Management

- Mechanisms for implementing memory management
  - physical vs. virtual addressing
  - base/limit registers
  - partitioning, paging, segmentation
- Internal and external fragmentation
Paged Virtual Memory

- Virtual address space
- Page faults
- Demand paging  
  - don’t try to anticipate
- Page replacement  
  - local, global, hybrid
- Locality  
  - temporal, spatial
- Working set
- Thrashing
- What is the complete set of steps for handling a page fault  
  - start to finish?
Page replacement algorithms

- Belady’s – optimal, but unrealizable
- FIFO – replace page loaded furthest in the past
- LRU – replace page referenced furthest in the past
  - approximate using PTE reference bit
- LRU Clock – replace page that is “old enough”
- Second chance (two-level FIFO due to lack of hardware support required for LRU clock)
- Working Set – keep the working set in memory
- Page Fault Frequency – grow/shrink number of frames as a function of fault rate
Multi-level page tables, TLBs

• How to reduce overhead of paging?
  – how do multi-level page tables work?
  – what problem does TLB solve?
  – how are they managed?
    • software vs. hardware managed

• Page faults
  – what is one? how is it used to implement demand paging?
  – what is complete sequence of steps for translating a virtual address to a PA?
    • all the way from TLB access to paging in from disk
  – cache organization and VM interaction

• MM tricks
  – shared memory? Mapped files? copy-on-write?
Processes

• What is a process? What does it virtualize?
  – differences between program, process, thread?
  – what is contained in process?
    • what does PCB contain?
    • PCB vs. address space
  – state queues?
    • which states, what transitions are possible?
    • when do transitions happen?

• Process manipulation
  – what does fork() do? how about exec()?
  – how do shells work?

• Inter-process communication (IPC)
  – “command line args,” pipes, signals, shared memory
  – shells

© 2017 Gribble, Lazowska, Levy, Zahorjan, Zbikowski
Threads

• What is a thread?
  – why are they useful?
  – what’s the address space look like?
  – TCB vs. PCB
  – user-level vs. kernel-level threads?
    • performance implications
    • functionality implications

• How does thread scheduling differ from process scheduling?
  – what operations do threads support?
  – what happens on a thread context switch? what is saved in TCB?
  – preemptive vs. non-preemptive scheduling?
  – scheduler activations
Processor Scheduling

• Long term vs. short term
• When does scheduling happen?
  – job changes state, interrupts, exceptions, job creation
• Scheduling goals?
  – maximize CPU utilization
  – maximize job throughput
  – minimize \{turnaround time | waiting time | response time\}
  – batch vs. interactive: what are their goals?
• What is starvation? what causes it?
• FCFS/FIFO, SPT, SRPT, priority, RR, MLFQ, CFS (completely fair scheduler)
Synchronization

• Why do we need it?
  – data coordination? execution coordination?
  – what are race conditions? when do they occur?
  – when are resources shared? (variables, heap objects, …)

• What is mutual exclusion?
  – what is a critical section?
  – what are the requirements of critical section solutions?
    • mutex, progress, bounded waiting, performance
  – what are mechanisms for programming critical sections?
    • locks, semaphores, monitors, condition variables
Locks

• What does it mean for acquire/release to be atomic?
• how can locks be implemented?
  – spinlocks? interrupts? OS/thread-scheduler?
  – test-and-set?
  – limitations of locks?
Semaphores and Monitors

• Semaphores
  – basic operations: wait vs. signal?
  – difference between semaphore and lock?
  – when and how do threads block on semaphores? when do they wake?
  – bounded buffers problem
    • producer/consumer
  – readers/writers problem
  – how is all of this implemented
    • moving descriptors on and off queues

• Monitors
  – the operations and their implementation
Non-blocking Synchronization

• What does it mean to be “non-blocking”? 
• Why might you want it? 
• Compare-and-swap semantics 
• “same value” problem and solution approach 
• General idea of implementation of a FIFO
Deadlock

- static prevention, dynamic avoidance, detection/recovery
- tradeoffs among these
- graph reducibility
- approaches
  - Hold and wait
  - Resource ordering
  - Banker’s algorithm
  - Detect and eliminate
Disks

- Physical (spinning) disk structure
  - platters, surfaces, tracks, sectors, cylinders, arms, heads
- Disk interface
  - how does OS make requests to the disk?
- Disk performance
  - access time = seek + rotation + transfer
- Disk scheduling
  - how does it improve performance?
    - FCFS, SSTF, SCAN, C-SCAN?
- Implications of solid state drives
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 does path name translation work?

• ACLs vs. capabilities
  – matrix
  – advantages and disadvantages of each
File system data structures

• General strategies?
  – contiguous, linked, indexed?
  – tradeoffs?

• What is a Unix inode?
  – how are they different than directories?
  – how are inodes and directories used to do path resolution, and find files?

• Everything about the Unix File System (UFS)
FS buffer cache

• What is a buffer cache?
  – why do OS’s use them?

• What are differences between caching reads and writes?
  – write-through, write-around, write-back/write-behind?
  – read-ahead?
FFS, JFS, LFS

• What is FFS, how specifically does it improve over original Unix FS?
• How about JFS, what is the key problem that it solves, what are the basic ideas?
  – Define “failure atomicity”.
• How about LFS, what are the basic ideas, when does it yield an improvement, when does it not?
RAID

• Basic concepts of RAID
  – stripe files across multiple disks to improve throughput
  – compensate for decreased reliability with parity/ECC

• Software vs. hardware implementation

• Sources of improvement among RAID-0, RAID-1, and RAID-5

• RAID vs. backup (they are different!)
Virtual Machine Monitors

- Basic concepts of VMM’s
- In some detail, what is the relationship between an application, the guest OS on which it runs, the VMM, and the hardware?
  - How does control transfer appropriately?
  - How do reconcile the fact that both the apps and the guest OS’s are running in user mode?
  - Be able to trace the handling of a syscall
- Binary translation
- Ways in which hardware implementations have been evolving to improve efficiency of VMMs
Projects

• You’re responsible for understanding all aspects of the projects!