CSE 451
Final Round
Review

Processes

- What is a process? What does it virtualize?
  - differences between program, process, thread?
  - what is contained in process?
    - what does PCB contain?
  - 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?

Threads

- What is a thread?
  - why are they useful?
  - 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?

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, …

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 sections?
    - 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?

Semaphores and Monitors

- Semaphores and Condition Variables
  - basic operations: wait vs. signal?
  - difference between semaphore and CV?
  - when and how do threads block on semaphores? CVs? 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

Deadlock

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

Memory Management

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

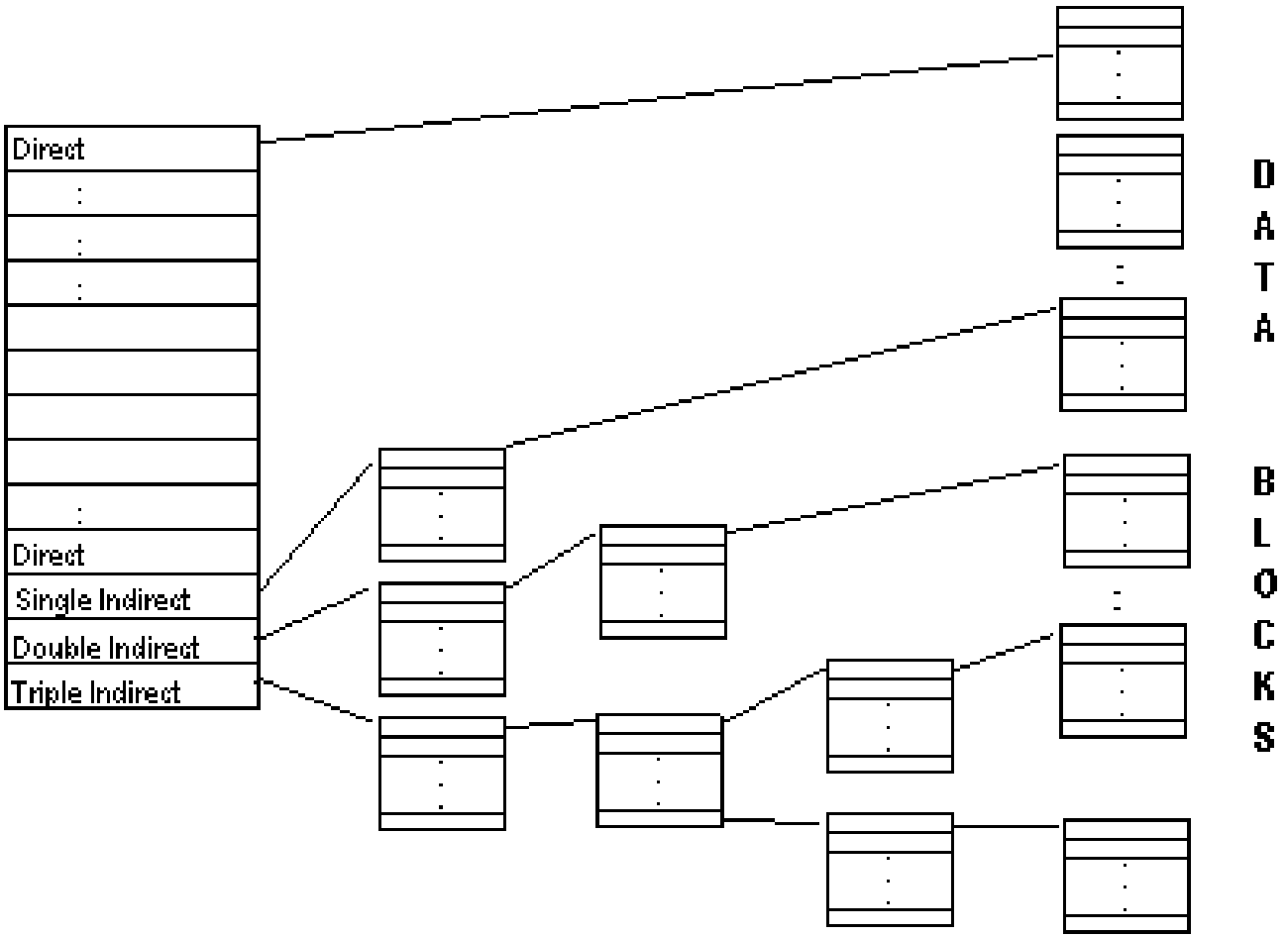
Review: virtual memory

- Linux uses both paging and segmentation
  - Paging allows independent address spaces for each process.
  - Segmentation sets up “kernel memory” and “user memory” segments – limited use in linux.
- How does copy-on-write work? What is it used for?
  - Writeable page mapped into multiple address spaces as one copy until first write.
  - Useful for fork – why?
- What is The OPT algorithm?
  - Optimal page replacement – evict page the won’t be needed longest into the future
- What is Belady's anomaly?
  - Bad property of FIFO – fault rate can increase with more allocated frames
- LRU has great performance but is inefficient in practice.

Review:

- Consider a modern desktop computer on which the hard disk is spinning. What is the most significant delay in reading from a 4K byte file that has not been accessed in a long time?

Unix File System



Source: wikipedia

Review: file systems

- Given 4k data blocks, a 8 entry data table with 6 direct entries, one single-indirection entry, and one double-indirection entry, what is the maximum file size?

FS cont.

- Explain the benefit of cylinder groups introduced in the BSD FFS?
  - ...
- What problems were JFS addressing and how?
  - ...

Last slide

- You have (almost) completed OS.

  It was fun, right?