Lecture 9-

Mesa Vs Hoare Monitors

Mesa Monitors use while() for locks because need multiple checks to make sure no dequeue after if statement.

Reader/Writer

Readers wait until no writers and writers wait until no readers or writers. Only one writer, with no reader at same time. Can be multiple readers. Only one lock I think.

Scheduling-

First in first out (fifo) finishes each task as arrives.

Response time/ throughput/ fairness

Round Robin scheduling- switch tasks every 100ms, context switch take at most 1ms

Lottery scheduling- give each process lottery tickets and choose at random. Every job gets at least one ticket, with shorter jobs getting more so they can probability get picked and finish faster. Adding or deleting jobs changes proportionality independent of priority of each.

Evaluate scheduling algorithm by deterministic loads, queueing loads, simulations in real world.

Apps that have short bursts get high priority because they finish fast without taking too much away from longer apps. Apps with lots of computations get low priority.

Shortest Job First, Shortest Remaining Time First minimize average response time/can cause starvation if many small jobs.

Can change future algorithm based on past data.

Real time scheduling used to predict and minimize worst case scenarios (either hard deadlines, or soft)

Starvation-thread waits indefinitely / Deadlock-Circular waiting for resources

Deadlock example-two lane road goes down to one lane

Requirements for deadlock – mutual exclusion/ threads can hold a resource while waiting for another / no preemption / circular wait

Can handle deadlock by having detection of deadlock and some way to force tasks out of it

Can ensure system never enters deadlock by monitoring socks and denying acquisitions that may cause deadlock

Bankers algorithm makes sure to avoid choices that will cause deadlock.

Memory Multiplexing – Protects access to private memory between processes, Contols overlap between threads so they don’t collide in physical memory, and translates access from virtual to physical space.

Memory Segmentation-

Virtual space has holes in it that don’t matter, heap and stack grow as allocated, segmentation stored in CPU because small.

Segmentation must fit variable sized chunks into physical memory, may move processes to fit everything, and can fragment to cause wasted space.

Fragmentation solved with pages. Page tables reside in physical memory.

Virtual address maps to