Exam 1 Material

1. Operating system types, hardware structures, and software structures.

2. Processes, including process creation, process communication (i.e., shared memory and message queues), and process scheduling. You may be asked to read and/or write simple C code using fork(), exec(), and wait().

3. Threads and thread management, and their relationship to processes.

4. Process synchronization, including mutual exclusion algorithms, semaphores, and monitors. You should understand these techniques and how to apply them to problems similar to bounded buffer, readers/writers, and dining philosophers. You may be asked to read and/or write pseudo­code programs and comment on their appropriateness as solutions based on the criteria of mutual exclusion, progress, and bounded wait.

5. Deadlocks, the four necessary conditions for deadlock, and identifying when a system is or is not in

deadlock.

Additional

Peterson’s Algorithm

Mutex Locks

Implicit Threading

OpenMP

Synchronization, Critical-section

Parallel Processing Techniques

IPC

2^4 = 16 processes // the for loop has 4 iterations

Speedup <= 1/ (S + ((1-S)/N)) s is portion that is serial with n processing cores

Fork returns 0 to newly created child process. Positive value to parent or caller, negative if failed

Getpid returns process id of this process

thread create spins up a thread given a pid and function to execute

Shortest job first will ideally have best run time as a scheduler. When a process is admitted an interrupt is done, and a decision is made.

Peterson: ME if other process is not in CS Pi gets in Pj is prevented from entering. If both want in a same time only one will get in. Progress. If Pj is not trying to get in Pi always gets in. If both trying to get in only one get in. Bounded. If both are trying to get in, the other waits a bounded amount of time.

Conditions necessary for deadlock: 1. ME. At least one of resources must require ME access. 2. Hold&wait. Every process must request and probably wait for a resource while holding one. 3. Non preemption. Resource cannot be preempted cheaply. 4. Circular wait. Cycle of hold and wait. Solutions; detect &recover, prevention, and dynamic rules based on system state.

One lane bridge: Eastbound. Request w, use w, request e, release w, use e, release e. Westbound. Request e, use e, request w, release e, use w, release w.

Dining philosophers. Implemented using shared semaphore. 1. Loop. 1.1 wait(chopstick[i]) wait(chopstick[i+1%5]) 1.2. CS 1.3. signal(chopstick[i]) signal(chopstick[i+1%5])

ReadersWriters. Writers. 1. Loop. 1.1.wait(wmutex) 1.2.write object 1.3.signal(wmatrix) 1.4. RS Readers. 1. Loop. 1.1.wait(rmutex) when writer wants access no new readers can access. Readers have access as long as the other readers can access.