Joseph Carrillo

CST-321

Operating System Fundamentals

Mutexes & Semaphores

5/29/22

Semaphores: Example (Server Lobby/Game Lobby)

Detailed Description of Program:

Server Lobby where there are total of 10 users in que waiting to join/enter the server lobby, using sem\_wait. Once the 10 users are accepted into the que the users enter the que and stay in the server somewhere between 1-10 seconds then exit. Sem\_post increments the value and allows any other users to join the server.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Mutexes: Example Scenario of Mail that gets distributed & Mutex Locks

Description: The mutex lock is executed by the pthread\_mutex\_lock function to help process the waiting of a lock if it is not available, the taking of a lock and once a lock is already taken Mutexes helps prevent from other threads from executing the same line of code, mutexes ensure the certainty of checking to see what thread has taken up a mailer locker

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Pros & Cons of Mutexes and Semaphores

A: Mutexes is a locking mechanism that allows for the synchronization of gaining access to a resource. One thread can obtain the mutex at a time and is useful for not allowing multiple threads to process that same operation. The con is that it can lead to starvation. In which the implementation can be in a busy, waiting state that wastes CPU time. Semaphores saves resources because there is no wasting time on waiting. The threads that are accessing the critical section are in que and can access the priority section once the thread is de-qued. Semaphores must be implemented in the correct way with utmost accuracy to prevent deadlocks