Faculty of Engineering and Applied Science

SOFE 3950 Operating Systems

Tutorial Activity 5/6

Group Member 1

Name: Daniel Nucci
Student ID: 100655384

Group Member 2

Name: Avdon Racki
Student ID: 100661246

Date: Monday, Feb 10th, 2020

Conceptual Questions

1. What is fork(), how does it differ from multi-threading (pthreads)?

A fork operation will result in a new process being created. This new process will inherit the parent process' code, however the memory and resources of the processes will not be shared. Threads exist within a process and a process can have multiple threads. Threads run in parallel and share the same memory and resources as the host process. Threads require the use of mutex/semaphores in order to ensure data access occurs sequentially and data is never accessed by more than one thread at a time.

2. What is inter-process communication (IPC)? Describe methods of performing IPC.

A process can either be independent or co-operating. Co-operating processes communicate with other co-operating processes and can share memory and pass messages. If multiple processes wish to communicate with one another via memory sharing, the implementation depends on how the programmer chooses to go about it.

One issue that arises with memory sharing is the producer/consumer problem where one process produces data and stores it in shared memory and one process consumes that data. Issues can occur when the producer process hits a buffer limit and the consumer process does not read the data.

If multiple processes communicate with one another via message passing instead of memory sharing, they can do so by utilizing the send() and receive() functions after establishing a communication link.

3. Provide an explanation of semaphores, how do they work, how do they differ from mutual exclusion?

A mutual exclusion is essentially a locking mechanism and restricts access to a shared resource while a thread has a key. A mutual exclusion is used when dealing with access to a single shared resource. A semaphore buffer can be split into multiple lanes and multiple threads can read/write to each lane. For example, if you have a 4KB buffer you can split that buffer into 4x1KB lanes and multiple threads can each use a lane to store/read data.

4. Provide an explanation of wait (P) and signal (V).

Both wait (P) and signal (V) are atomic. This means that no other operation is performed during the read/write and update operations. By default when a semaphore is created it is initialized to 1 which means that the shared resource is able to be accessed by a process. When a process accesses a shared variable the semaphore is set to 0. When one process enters a critical section it executes the wait (P) operation on the semaphore which makes this change occur. Another process cannot enter the critical section until the current process accessing the critical section calls the signal (V) operation on the semaphore which indicates that it is done with the critical section.

5. Research the main functions used for semaphores in and explain each function.

In C the main functions for semaphores include: sem_wait() which locks the semaphore, sem_post() which releases and unlocks the semaphore, sem_init() which creates a semaphore, and sem_destroy() which deletes a semaphore.

Application Questions

See attached code.