TUTORIAL #6: POSIX Threads Part II

Faculty of Engineering and Applied Science
Operating Systems SOFE-3950U

74171 Group 11
Awais Ahmad - 100785259
Mithusan Arulampalam - 100787123
Jawad Kohja - 100786085
Mohammad Mohammadkhani - 100798165

Date: March 15, 2023

1. **What is fork(), how does it differ from multi-threading (pthreads)?**
   Forking splits the current process into a new child process that inherits a lot of code and data from the parent. In many systems, only one process can run at a time. Multi-threading creates a lightweight process, where multiple threads can execute in parallel since they share memory and other resources.

2. **What is inter-process communication (IPC)? Describe methods of performing IPC.**
   Interprocess communication is a way to allow processes/ threads to communicate with each other. Some methods to allow for IPC are **ordinary pipes** (one end is read and one end is write), **named pipes** (allows both processes to read and write), **shared memory** (an area of memory shared between processes), and **message passing** (messages are sent to a queue to be read by another process).

3. **Provide an explanation of semaphores, how do they work, how do they differ from mutual exclusion?**
   A semaphore is a synchronization tool that provides sophisticated ways for different processes to synchronize their activities. It's used to control access to a shared resource, or to notify threads when a certain resource is available. They use the wait() and signal() functions to synchronize the processes. The difference between a semaphore and mutual exclusion is that a mutex uses a locking mechanism while semaphores use a signaling mechanism. Additionally, a mutex allows threads to access a variable one at a time while a semaphore allows threads to access multiple instances of a resource.

4. **Provide an explanation of wait (P) and signal (V).**
   - **wait(P):** used when a process wants to use a resource. This decrements the value of the semaphore. If the value of the semaphore is 0, the process that calls the wait(P) will be blocked until there is an available semaphore.
   - **signal(V)**: used when a process releases a resource. This increments the value of the semaphore.

5. **Research the main functions used for semaphores in <semaphore.h> and explain each function.**
   - **sem_close**: close the named semaphore.
   - **sem_destroy**: destroy the unnamed semaphore.
   - **sem_getvalue**: get the current value of the semaphore.

- **sem_init**: initialize an unnamed semaphore. the value of it is the int value passed to the function.
- **sem_open**: initialize and open a named semaphore.
- **sem_post**: unlock the semaphore. If the value returned was positive, then there were no threads that were blocked. If it is 0, one blocked thread will be allowed to return from its sem_wait() call.
- **sem_trywait**: locks the semaphore if it is not currently locked. The semaphore remains locked until the sem_post() function is called.
- **sem_unlink**: remove the named semaphore. if any processes are using the semaphore, removal is postponed until the references have been destroyed.
- **sem_wait**: locks the semaphore until sem_post() function is called to unlock it.

.