Lab Exercise 4 – Semaphores

The purpose of Lab Exercise 4 is to start working with System V semaphores on Linux.

Semaphore operations and Linux semaphore functions:

The two semaphore operations, P() and V(), described in Chapter 14 of the Linux book are equivalent to wait() and signal(), respectively, described in class. There are three Linux semaphore functions:

- **semget ()**
- **semop()**
- **semctl()**

semget() is used to create a new semaphore or obtain the key of an existing semaphore.

semop() is used to perform operations on the semaphore. Go over the description on semop(). Identify the operation/value needed for the P operation, i.e., wait(), and the V operation, i.e., signal().

semctl() is used to control information in a semaphore (can be used in initializing a semaphore or for removing a semaphore). Why is removing a semaphore necessary?

Read the description of each of these functions from the Linux book.

Try it out – Semaphores, sem1.c

Compare the two semaphore operations P() and V(), implemented with semaphore_p and semaphore_v in the example, respectively. What is the difference between those two functions? Which system function is used in P() and V()?

Run the program and see how it works. **Walk through** the code and go over those semaphore operations and understand how each function works.

Applications

1. A semaphore can be initialized to a different value, e.g., k in the circular buffer for the producer/consumer problem. If an initial value of k is needed for a semaphore, where in the code is the right place to initialize the value to k?

2. Semaphores can be used for synchronization. For instance, consider two processes, P1 and P2. P1 has to wait until P2 finishes statement S2 before P1 can execute statement S1. Implement the example using the system semaphore functions and the functions provided by the Linux book.

S1 and S2 can be just printf() statements. Run the program several times and see if the results are consistent, i.e., no race condition is happening.

Submit an archive (using the command tar) containing your Makefile and all your source files (both *.c and *.h) on cuLearn