Brian Duenas

CSE 460

Lab 4

20 points Total

1. **Shared Memory**

**Q:** Write a brief description on the usage of each semaphore.

**A:**

*Shmget()* : returns the identifier of the System V shared memory segment associated with the value of the argument key. ( shared memory get )

*Shmat()* : attaches the System V shared memory segment identified by *shmid* to the address space of the calling process. ( shared memory attach )

*Shmdt()* : detaches the shared memory segment located at the address specified by *shmaddr* from the address space of the calling process. ( shared memory detach )

*Shmctl()* : performs the control operation specified by cmd on the System V shared memory segment whose identifier is given in *shmid*. ( shared memory control )

**Q:** Type in some text at the terminals. What do you see? What text you enter will terminate the programs? Explain what you have seen.

**Output:**

****

****

**A:** When typing into the terminal with the program *shared2* running the text appears in the terminal window with *shared1* running. Typing “end” will terminate both programs. Both programs are accessing the same memory space so when *shared2* writes (produces) program *shared1* reads the change (consumes).

**Code for *shared1.cpp*:**

****

**Code for *Shared2.cpp*:**

****

2. **POSIX Semaphores**

**Q:** Try the server-client example and explain what you observe. You have to start the server first why?

**Outputs:**

****

****

**A:** The server starts and writes to memory. The client reads from memory and signals to server that it has finished reading. Server must starts first because it creates the memory space with the flag *O\_CREAT*, while the client opens a created memory space.

**Q:** Modify the programs so that the server sits in a loop to accept string inputs from users and send them to the client, which then prints out the string.

**Code for *server.cpp*:**

****

**Code for *Client.cpp*:**

****

3. **XV6 System Calls**

**Q:** Modify the files and show the output for *ps*.

**Output:**

****

**Q:** Modify ***cps****()* in *proc.c* so that it returns the total number of processes that are SLEEPING or RUNNING.

**Code:**

****

**Output:**

****