Advanced Operating Systems Lab Assignments (CS 401)

M. Tech. CS, 2018

Note: *Please arrange all the programming assignments according to the given list. You need to explain in detail about the assignments/problems /methods /solutions.*

Part-I

1. Shell commands and shell scripting

   (a) Explain about the following commands along with parameters or arguments: **date, cal, echo, man, ls, pwd, mkdir, cd, rmdir, cat, sort, mv, cp, rm, wc, head, tail, more, pipe, grep, tr, chmod, chwon, chgrp, ps, pstree, top, pgrep, renice, kill, xkill.**

   (b) Write a shell script to generate the Fibonacci series

2. Write a C/C++ program that creates a process using the fork() call. Use the getpid() and getppid() calls to print the child and parent id in created child process, and return value of fork call in parent process.

3. Write a C/C++ program to create a process using the exec() call. What is the difference between a fork() and an exec() call?

4. Write a C/C++ program to find out the number of processes and their states, like CPU time, submission time, memory size of the process etc. at any instant. Store the information in a table structure for further processing.

5. Write a C/C++ program to fiind the number of processes (together with all process details) running on the system using system calls. Additionally, you may use fork() system call to create more processes.

6. Write a C/C++ program to find the average size of processes.

Part-II

1. (Implement using C/ C++ program) : Given a set of processes (P1, P2, ..., Pn) and resources (R1, R2, ..., Rn), at any instant, the set of requests (Pi to Rj) and the set of assignments (Ri to Pj) are given:

   (a) Find out that no single resource is assigned to multiple processes.

   (b) Find out that the processes is not making repeated requests for the same resource.

2. Implement usin C/C++ programming : The parent send signals to child using the *pid* and kill(). The child picks up these signals with signal() function and calls appropriate functions to handle it.

   - The parent send the signal when you type Ctrl+C from your keyboard.
   - All signals types are defined in **signal.h** header file.

- Implement the program for both the signal SIGINT(Ctrl+C) , SIGQUIT(Ctrl+D)
- The communication between perent and child processes is done using kill() and signal(), fork() system call.

3. Two process process P1 and process P2. For this you need to write two programs that run on two diffrent terminal on your system. The process P1 should capture the *interrupt* i.e. *Ctrl+C* from the keyboard you typed and send the interrupt signal to the process P2. When process P2 receives the signal it shoud display it along with the process id of the sender process (since there may be many sender processes, so to check which process sends the signal we need to display the process ID's). As many times you type the *Ctrl+C* from the sender process terminal the receiver process P2 should display it .

   - All signals types are defined in **signal.h** header file.
   - For this program you may use *sigaction()* function or *signal()* function.
   - You can use the techniques used for inter process communications *i.e. message queue or shared memory or pipes etc.*

4. (IPC): Write a C/C++ program that creates two processes, a parent and a child where the child is created via the fork() system call. The parent reads the content of a file and sends them to child via Interprocess communication (IPC).The child receives the data send by the parent and write them to another file.

   Hints: You can use any one of the IPC mechanism like UNIX/LINUX pipes, Shared memory or message queue .

5. (IPC): Write two C/C++ programs that creates two processes (process1 and process2). The process 1 reads the content of a file and sends them to process 2 via Interprocess communication (IPC).The receiver process (process 2) receives the data send by the sender process (process 1) and write them to another file.

   Hints: You can use any one of the IPC mechanism like UNIX/LINUX pipes, Shared memory or message queue .

6. **Deadlock Detection**: Write a C/C++ program to detect the deadlock in a system. You might consider using simple data structures like single and two dimensional arrays to represent your system resources and processes .You are free to input the number of processes and resources, or use random number generator to initialize a snapshot of the state of the system.

7. **Producer/Consumer problem using shared memory and semaphores:** Each producer reads its input file one character at a time and writes it into a shared buffer.The consumer consumes characters from this buffer one at a time and writes them into output files. The size of the buffer is finite. *Use* ***semaphores for synchronization*** *between producer and consumer processes.* There should be three different files two input files for producer and one output file for consumer as shown in the figure.

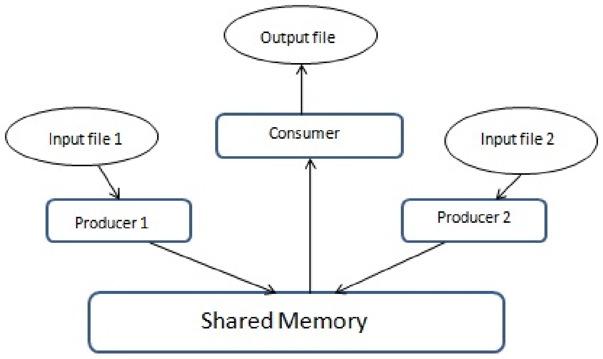

Figure 1: Producer / consumer problem diagram

8. Implement using C/C++ programming : Create 'n' number of virtual memory partitions where n is the total number of processes. Each partition is of random size between minimum process size to 20% more than the maximum process size.

9. Implement using C/C++ programming : Find the total internal fragmentation and external fragmentation using

   - First-fit
   - Best-fit
   - Worst-fit

References: You may follow the below references for some of the above assignments

1. `https://users.cs.cf.ac.uk/Dave.Marshall/C/`

2. `http://beej.us/guide/bgipc/html/single/bgipc.html`

3. BOOK: Beginning Linux Programming ,Neil Matthew & Richard Stones.