

# LBS College of Engineering Kasaragod

## Department of Computer Science and Engineering

### PCCSL407 OPERATING SYSTEMS LAB (L: T:P: R -0:0:3:0)

#### Lab cycle

1. Familiarisation with basic Linux programming commands: ps, strace, gdb, strings, objdump, nm, file, od, xxd, time, fuser, top
2. Familiarization of /proc file
  - Use /proc file system to gather basic information about your machine:
    - (a) Number of CPU cores
    - (b) Total memory and the fraction of free memory
    - (c) Number of processes currently running.
    - (d) Number of processes in the running and blocked states.
    - (e) Number of processes forked since the last bootup. How do you compare this value with the one in (c) above?
    - (f) The number of context switches performed since the last bootup for a particular process.
  - Write a simple program to print the system time and execute it. Then use the /proc file system to determine how long this program (in the strict sense, the corresponding process) ran in user and kernel modes.
3. Process creation
  - Create a new process using a fork system call. Print the parent and child process IDs. Use the pstree command to find the process tree for the child process starting from the init process.
  - Write a program to add two integers (received via the command line) and compile it to an executable named "myadder". Now write another program that creates a new process using a fork system call. Make the child process add two integers by replacing its image with the "myadder" image using execvp system call.
  - Create a new process using a fork system call. The child process should print the string "PCCSL407" and the parent process should print the string "Operating Systems Lab". Use a wait system call to ensure that the output displayed is "PCCSL407 Operating Systems Lab"
4. Interprocess Communication
  - Using Pipe – Evaluate the expression . The first process evaluates . The second process evaluates and sends it to the first process which evaluates the final expression and displays it.
  - Using Message Queue - The first process sends a string to the second process. The second process reverses the received string and sends it back to the first process. The first process compares the original string and the reversed string

received from the second one and then prints whether the string is a palindrome or not.

- Using Shared Memory - The first process sends three strings to the second process. The second process concatenates them to a single string (with whitespace being inserted between the two individual strings) and sends it back to the first process. The first process prints the concatenated string in the flipped case, that is if the concatenated string is "Hello S4 Students", the final output should be "hELLO s4 sTUDENTS"

#### 5. Multithreading

- Write a multithreaded program that calculates the mean, median, and standard deviation for a list of integers. This program should receive a series of integers on the command line and will then create three separate worker threads. The first thread will determine the mean value, the second will determine the median and the third will calculate the standard deviation of the integers. The variables representing the mean, median, and standard deviation values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited.

#### 6. CPU Scheduling

- Given below is a table containing a list of processes, their CPU burst times (integral values), arrival times, and priorities. Simulate FCFS, SRTF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 3 units) scheduling algorithms on the process mix. Compute the completion time, turn around time and waiting time of each processes and Determine which algorithm results in the minimum average waiting time (over all processes).

Process ID	Arrival Time	Burst Time	Priority
P1	0	7	2
P2	2	4	4
P3	4	1	1
P4	5	4	3
P5	6	2	2

7. Use semaphores to solve the readers-writers problem with writers being given priority over readers.

8. Simulate the banker's algorithm to determine a safe execution sequence. Consider a system with n processes and m resource types. The system is described by the following data structures:

- Allocation Matrix – resources currently allocated to each process
- Maximum Matrix – maximum resources each process may request
- Available Vector – resources currently available in the system

Using Banker's Algorithm, determine whether the system is in a safe state. If the system is safe, print a safe execution sequence of processes. Otherwise, indicate that the system is deadlocked or unsafe.

### Input Specification:

- Number of processes  $n = 5$  (P0, P1, P2, P3, P4)
- Number of resource types  $m = 3$  (A, B, C)

Allocation =	Maximum =	Available = 3 3 2
0 1 0	7 5 3	
2 0 0	3 2 2	
3 0 2	9 0 2	
2 1 1	2 2 2	
0 0 2	4 3 3	

### Output Specification

- Display the Need matrix
  - Display whether the system is SAFE or UNSAFE
  - If safe, print the safe execution sequence
9. Implement the deadlock-free semaphore-based solution for the dining philosopher's problem.
  10. Simulate the address translation in the paging scheme as follows: The program receives three command line arguments in the order
    - size of the virtual address space (in megabytes)
    - page size (in kilobytes)
    - a virtual address (in decimal notation)
 The output should be the physical address corresponding to the virtual address in <framenumbers, offset> format. You may assume that the page table is implemented as an array indexed by page numbers. (NB: If the page table has no index for the page number determined from the virtual address, you may just declare a page table miss!)
  11. Simulate the FIFO, LRU, and optimal page-replacement algorithms as follows: First, generate a random page-reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Assume that demand paging is used. The length of the reference string and the number of page frames (varying from 1 to 7) are to be received as command line arguments.
  12. Simulate the SSTF, LOOK, and CSCAN disk-scheduling algorithms as follows: Your program will service a disk with 5,000 cylinders numbered 0 to 4,999. The program will generate a random series of 10 cylinder requests and service them according to each of

the algorithms listed earlier. The program will be passed the initial position of the disk head (as a parameter on the command line) and will report the total number of head movements required by each algorithm.