OPERATING SYSTEMS (CS F372)
ASSIGNMENT
SEM I 2021-2022

WEIGHTAGE – 15%, MARKS – 45, TYPE – OPEN BOOK

MARKS DIVISION – PROBLEM: 37 MARKS, VIVA (TO BE CONDUCTED DURING DEMO): 8 MARKS

Problem Statement: [37 Marks]

In this problem, you will emulate two process scheduling algorithms for different workloads and analyse their performance. **The program is to be written in C and should be executable on a Linux platform**.

You are required to create one master process, M which spawns 3 child processes, C1, C2, and C3.

- C1 is a compute-intensive process which adds **`n1`** numbers in the range 1 to 1 million.
- C2 is an I/O intensive process which reads **`n2`** numbers (range from 1 to 1 million) from a text file and prints them to the console. Assume that each number is present in a separate line in the file. After printing all the numbers, C2 sends the message "**`Done Printing`**" to M using a pipe.
- C3 is both compute and I/O intensive which reads **`n3`** numbers (range from 1 to 1 million) from a text file and adds them up.
- C1 and C3 communicate the results of their operations to M using pipes. M prints these results on the console.
- Note that 3 separate pipes are to be used. The text files along with the contents will be provided during the demo. Create your own file while testing your code.

The following is a sample file structure. Assume the file to have .txt extension.

```
23
35
2222
56
8892
239085
```

In M, emulate the scheduling algorithms FCFS and Round Robin (RR) with a fixed time quantum. For RR, the time quantum is given as user input. Assume the order of process creation is C1, C2 and C3. C1, C2 and C3 are to be scheduled only after all the 3 processes have been created by M. The emulation of scheduling is to be done using the **`sleep()`** function. M will use shared memory segments with C1, C2 and C3 to communicate if C1, C2, C3 should sleep or wake up to perform their designated tasks. Within each of C1, C2 and C3, use one thread to do the task (mentioned earlier) and another to monitor communications from M and accordingly put the task thread to sleep, or wake it up. Assume a uniprocessor environment. At a time only one of C1, C2 or C3 can be awake. Note that, when M tells a process (either C1 or C2 or C3) to sleep, only the task thread should sleep. The monitor thread should remain awake in order to monitor communications from M. You can decide the frequency at the which the monitor thread checks for communication from M or can design some form of notification mechanism. Mention this clearly as a comment in your source file as well as during the demo.

The choice of the scheduling algorithm will be given as user input. **`n1`**, **`n2`** and **`n3`** constitute the process workload and are also given as user inputs. You should be able to repeat for different sizes of workload for performance analysis. The values of **`n1`**, **`n2`** and **`n3`** will range between 25 and one million.

Expected Output:

1. Output of the tasks performed by C1, C2 and C3 along with the time required to perform the individual tasks. Note that for C1 and C3, M will print the outputs and the time. C2 will print its own output along with the time required.
2. The sequence in which the 3 processes are scheduled along with when one process starts execution and when the context switch to the next process takes place. For eg., If C1 is scheduled at t1, then C2 is scheduled at t2, C3 is scheduled at t3, you should print on the console
   C1 starts at t1
   C2 starts at t2
   C3 starts at t3
3. Turnaround time and waiting time for each of C1, C2 and C3 using the different scheduling algorithms.
4. Your analysis on the performance of the different algorithms for different workloads. This is to be written in a separate text file.
5. Graph plots of turnaround time vs workload size and waiting time vs workload size for the different scheduling algorithms for the 3 child processes.

Tasks to be Performed and Marks Division:

1. Process and thread creation (5 marks)
2. Inter Process Communication (8 marks)
3. Process Scheduling (FCFS, RR) (12 marks)
4. I/O, Compute, I/O + Compute Intensive tasks performed by individual the task threads and monitoring activity performed by each of the monitor threads (12 marks)
5. Demo and Viva - 8 marks

List of Deliverables:

1. Code with proper comments (In the header part of the code, mention the name and IDs of the group members)
2. Performance Analysis (to be written in a separate text file)
3. Graph plots (graphs can be plotted using any tool)

Submission Procedure:

To be done in Section L for the course

Create a zipped folder containing all the deliverables and submit the zipped folder on CMS in the **Lecture Section L in the Assignment section**. Do not submit individual files on CMS. Do not attempt to submit the assignment either on L1 or L2.

The zipped folder should be named as per the Group No. that will be assigned to each group and will be notified later.

ONLY ONE SUBMISSION SHOULD BE MADE PER GROUP. SO, MAKE SURE ONLY ONE GROUP MEMBER SUBMITS THE ASSIGNMENT AFTER THE SOLUTIONS HAVE BEEN FINALIZED. NOTE THAT NO SUBMISSION VIA EMAIL WILL BE ACCEPTED.

Note that while submitting the assignment on CMS, you will have to explicitly click on the Submit button. DO NOT KEEP YOUR SUBMISSION SAVED AS DRAFT.

Deadline:

The deadline for the submission is 15th November 2021. Any request for deadline extension will not be entertained under any circumstances.

NOTE THAT WITHOUT ANY SUBMISSION, DEMOS WILL NOT BE TAKEN. SO EVEN IF YOU ARE ABLE TO PARTIALLY COMPLETE THE ASSIGNMENT, YOU SHOULD

SUBMIT ON CMS. IF A GROUP FAILS TO MAKE A SUBMISSION BY THE DEADLINE, THAT GROUP WILL NOT BE ALLOWED FOR DEMOS.

Doubt Clarification:

A discussion forum has been created on Piazza the link to which will be shared with you. In case of any doubts, please get them clarified by posting your queries on Piazza. The faculty members will answer your queries.

Plagiarism Policy:

The source code submitted by each group will be run through a code plagiarism checker. If the similarity indices of the submitted source codes exceed certain thresholds (which will be decided and notified later), marks penalty will be imposed on the concerned groups irrespective of the correctness of the submitted solutions. Note that merely renaming variables and keeping the flow of logic same including the order of declaration of variables and data structures does not make your program different from another one and ultimately adds up to the similarity index. Remember that if 2 groups are individually writing a program, it is highly improbable that a large portion of the code will be same. If you are attempting the assignment honestly, you will not have to worry about the similarity percentages.

****