

Operating Systems Lab (CL2006)

Date: December 23th, 2024

Lab Instructor(s)

Mr. Usman Anwar, Ms. Saba Tariq,
Ms. Haiqa Saman, Mr. Hashir,
Ms. Sukhan Amir, Ms. Samra Bhatti,
Mr. Hassan Raza, Mr. Zulqarnain,
Ms. Hooria Najeeb, Mr. Affan Malik

Final Exam

Total Time: 150 Minutes

Total Marks: 60

Total Questions: 02

Semester: Fall-2024

Campus: Lahore

Dept: CS, DS, SE

Student Name	Roll No	Section	Student Signature
--------------	---------	---------	-------------------

Instruction/Notes:

- Path: cactus1/xeon/Fall 2024/Saba Tariq/Operating Systems Lab Final/CS or DS or SE/your section
- If you find any ambiguity in a question, you can make your own assumption and answer the question accordingly by stating the same.
- Attempt all the questions.

CLO 2: Evaluate the commonly used mechanisms for scheduling of tasks and implement synchronization mechanisms like Semaphores, TSL, etc.

Question # 1: Implement a distributed task scheduler that uses shared memory and pipes for communication between the scheduler and worker processes. [30 marks]

Problem Statement:

You need to write a program that functions as a task scheduler. This scheduler will:

1. Use **shared memory** to maintain a **queue of tasks**.
2. Use **pipes** for communication between the scheduler and worker processes.
3. Each worker process should **pick a task** from the shared memory queue, execute it, and report the result back to the scheduler through pipes until there is no task left in queue.
4. The scheduler should **collect the results** from the pipes and **update the task status** in shared memory.

Detailed Steps:

- ✓ 1. Create shared memory for the task queue including 10 tasks in total. (1,2,3,4,5,6,7,8,9,10). 5 marks

- ✓ 2. Use `fork()` to create 3 worker processes.
- ✓ 3. Each worker process will read a task number from shared memory, prints task number assigned to it and write "0" in queue at its place. All process will execute in parallel. 10 marks
- ✓ 4. Each process will also use pipes to send a "Success" message to the parent process. 5 Marks
5. After receiving required number of "success" messages i.e. 10, parent process will **recheck shared queue** to see there is not task left and print a message before exiting itself. 5 marks
6. Use semaphores to **synchronize** access to the task queue. 5 marks

Note: Ensure proper synchronization and error handling.

CLO 2 : Implement solutions employing concepts of Processes and Threads tools/techniques.

Question # 2: Imagine you are tasked with designing a program to simulate a multiplayer gaming server. The idea is to allow multiple players to join, organize them into game rooms, and keep the game rooms running smoothly. Players will join and leave dynamically, and the system needs to handle these operations using both threads and processes. [30 Marks]

Here's what the program should do:

- ✓ 1. When the server starts, it should **initialize** a setup that shows all of 4 game rooms are available now. 2 marks
- ✓ 2. There are 10 players in total; the server should create a **child process** to represent each player. 5 marks
- ✓ 3. In a child process the player will tell which game room he wants to join (A hard code value Between 1 to 4). Depending upon that specific thread will be called. **Hint: Use Cases.** 5 marks
- ✓ 4. If that room is free it will be assigned to that player else he has to wait. 10 marks
- ✓ 5. Each game room will have a **room manager thread**. The thread will handle **periodic game updates** (every 2 seconds) this could be a simple terminal message to show the status. A room manager should keep running as long as there are players in the room. 5 marks
- ✓ 6. Players don't stay forever—they play for a **random amount of time** (somewhere between 3 and 7 seconds). After this, the player will leave the game, freeing up their slot for another player. 3 marks

Your task is to implement this system. You must use semaphores for synchronization to make sure there are no race conditions when managing shared resources, like the game room queue or player count. Proper cleanup of threads and processes is crucial when players leave or the server shuts down.

You must demonstrate:

1. Use of **fork()** to create player processes.
2. Use of **threads** to manage game rooms.
3. Synchronization using **semaphores** to avoid race conditions.

Write clean, readable code with comments to explain your thought process.