18

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

SEMESTER FINAL EXAMINATION **WINTER SEMESTER, 2022-2023**

DURATION: 3 HOURS **FULL MARKS: 150**

CSE 4501: Operating Systems

Programmable calculators are not allowed. Do not write anything on the question paper.

Answer **all 6 (six)** questions. Figures in the right margin indicate full marks of questions whereas corresponding CO and PO are written within parentheses.

---

1. a) What defines a process in an operating system, and how does it differ from a program? Provide a brief overview of various process states, and explain the transitions between these states using diagrams and examples. — 3 + 7 (CO1) (PO1)

   b) In the context of Inter Process Communication (IPC) in a distributed computing environment, describe a scenario where: — 5 × 2 (CO1) (PO1)

      i. Shared memory performs better than message passing

      ii. Message passing performs better than shared memory

      For each of the scenarios, explain how the choice of the IPC techniques depends on the specific characteristics of the computational problem.

   c) What resources are used when a thread is created? How do they differ from those used when a process is created? — 2 + 3 (CO1) (PO1)

2. a) A supercomputer is designed to perform a simulation task that consists of three distinct phases: preprocessing, computation, and postprocessing. The preprocessing phase is entirely sequential, the computation phase can be parallelized to a certain extent, and the postprocessing phase is again entirely sequential. The total execution time of the simulation task without any parallelization is 100 units of time. After careful analysis, it is determined that 20% of the entire task is preprocessing (sequential), 50% is computation (parallelizable), and the remaining 30% is postprocessing (sequential). — 5 + 5 (CO2) (PO2)

      i. Determine the speedup of the system when using 8 processors to parallelize the computation phase.

      ii. Discuss how the speedup and efficiency would change if the proportion of the computation phase that can be parallelized increases to 70%.

   b) Consider a multimedia application that requires efficient handling of both CPU-bound and I/O-bound tasks. The application performs real-time processing of multimedia data, such as decoding video frames and concurrently managing user inputs for interactive control. To optimize task processing, the system needs to map user-level threads to kernel-level threads. Which thread mapping model would you choose for this scenario? Provide justification, including reasons for not selecting the other models. — 10 (CO2) (PO2)

   c) Google's Chrome browser opens each new tab in a separate process. Would the same benefits have been achieved if Chrome opens each new tab in a separate thread? Explain your answer. — 5 (CO2) (PO1)

3. a) Write a thread program using the POSIX thread library. The program should include a 'control thread' that issues `START` and `STOP` commands to a 'processing thread'. Upon receiving the `START` command, the 'processing thread' should create and initiate a new 'worker thread'. The 'worker thread' is designed to print a single line 'The task has been started' and then wait in a loop. The 'processing thread' should be able to stop a 'worker thread' in the waiting loop upon receiving the `STOP` signal. — 10 (CO2) (PO3)

b) Write the program described in Question 3.a) using the POSIX library, this time implementing it with the concepts of processes and shared memory. (You are not allowed to use threads) — 10 (CO2) (PO3)

c) Consider the POSIX C program in the Code Snippet 1. Write the program's output. Your answer should be in the exact order as displayed by the compiler. — 5 (CO2) (PO2)
Assume that the process executing the program has a process ID of X. This process creates a child process with a process ID of $X + 1$, where X is the last four digits of your student ID. For instance, if the student ID is 200042019, X will be 2019.

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>

int main()
{
    pid_t pid, pid1;
    /* fork a child process */
    pid = fork();
    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid > 0) { /* parent process */
        pid1 = getpid();
        wait(NULL);
        printf("pid = %d\n",pid);
        printf("pid1 = %d\n",pid1);
    }
    else { /* child process */
        pid1 = getpid();
        printf("pid = %d\n",pid);
        printf("pid1 = %d\n",pid1);
    }
    return 0;
}
```

Code Snippet 1: A C program that creates a child proccess using POSIX library for Question 3.c)

4. a) Annually, immigrants seek U.S. Citizenship through a naturalization process, overseen by a judge and involving both immigrants and spectators. Immigrants need to wait in line, check in, and then sit down. At some point, the judge enters the building. When the judge is in the building, no one, including immigrants, can enter, and immigrants cannot leave. Spectators are allowed to leave during this time. Once all immigrants have checked in, the judge can confirm naturalization. After confirmation, immigrants can pick up their U.S. Citizenship certificates, and the judge leaves. Spectators may then enter, and after immigrants receive their certificates, they can leave. — 15 (CO4) (PO3)
Now considering each immigrants, spectators and the judge as separate thread design the synchronization mechanisms to ensure proper coordination among them, adhering to the specified task order. To implement this scenario, you can use Mutex Locks, or Semaphores, or both. Focus on writing synchronization aspects; providing a complete runnable code is not required.

b) The Four Cars (Junction) Synchronization Problem is a classic synchronization problem used in computer science and operating systems to illustrate issues related to concurrent processes. The scenario involves four cars arriving at a junction, each coming from a different direction. The challenge is to synchronize the movement of the cars to avoid collisions. One of your friends attempted the solution depicted in the Code Snippet 2, but it proved unsuccessful in achieving synchronization. At times, it resulted in a deadlock scenario. Using deadlock characterization, explain the reasons behind this deadlock occurrence. Propose modifications to address the synchronization problem and enhance the effectiveness of the solution. 10 (CO4) (PO2)

```
1  Semaphore mutex = 1;    // Binary semaphore for mutual exclusion
2  Semaphore carA = 0;     // Semaphore for Car A
3  Semaphore carB = 0;     // Semaphore for Car B
4  Semaphore carC = 0;     // Semaphore for Car C
5  Semaphore carD = 0;     // Semaphore for Car D
6
7  // Car A
8  wait(mutex);
9  signal(carA);
10 wait(carB);
11 signal(mutex);
12 // Critical Section for Car A
13 signal(carB);
14
15 // Car B
16 wait(mutex);
17 signal(carB);
18 wait(carC);
19 signal(mutex);
20 // Critical Section for Car B
21 signal(carC);
22
23 // Car C
24 wait(mutex);
25 signal(carC);
26 wait(carD);
27 signal(mutex);
28 // Critical Section for Car C
29 signal(carD);
30
31 // Car D
32 wait(mutex);
33 signal(carD);
34 wait(carA);
35 signal(mutex);
36 // Critical Section for Car D
37 signal(carA);
```

Code Snippet 2: A solution to the Four Cars (Junction) Synchronization Problem for Question 4.b)

5. a) What are the advantages of having different time-quantum sizes at different levels of a multilevel queuing system? 5 (CO1) (PO1)

b) Consider the following set of processes in the Table 1. The processes are scheduled using the Shortest Remaining Job First (SRJF) scheduling algorithm with exponential averaging for predicting the next CPU burst. 4 + 6 (CO3) (PO2)

Draw the Gantt chart. Calculate the average waiting time, response time, and turnaround time for each process using the SRJF scheduling algorithm with exponential averaging.

Table 1: Process table for Question 5.b)

Process	Arrival Time	Previous Predicted Burst Time (τ_n)	Previous Actual Burst Time (t_n)	α
P1	0	6	7	0.6
P2	1	5	4	0.4
P3	2	3	5	0.5
P4	3	2	4	0.3
P5	4	4	2	0.5

c) As a system architect for a cloud-based service, you are tasked with designing a scheduling mechanism to manage a diverse set of tasks. The system caters to both short-term interactive tasks, long-running background tasks, and computational tasks with varying processing needs. Additionally, there's a need to prioritize tasks based on their historical behavior, ensuring fair resource allocation. 3 + 4 + 3 (CO3) (PO2)

i. Explain why you would choose the Multi-level Feedback Queue (MLFQ) scheduling algorithm for this scenario.

ii. Discuss how MLFQ addresses the challenges posed by the diverse nature of tasks, varying processing times, and the need for adaptive prioritization based on task behavior.

iii. Justify how MLFQ aligns with the specific requirements of the cloud-based service.

6. a) Consider the operational dynamics of a banking system, where two functions play vital role: `deposit(amount)` and `withdraw(amount)`. These functions are passed the amount to be deposited or withdrawn from a shared bank account, which is jointly held by a husband and wife. In a concurrent scenario, the husband calls the `withdraw()` function while the wife calls `deposit()`. 4 + 6 (CO4) (PO2)

Describe how a race condition is possible in this scenario and suggest measures to prevent the race condition from occurring.

b) Consider a real-time system with three tasks: Task X, Task Y, and Task Z, having the following properties. 4 + 6 (CO4) (PO2)

- Task X (High Priority): Executes critical operations that require access to a shared resource. Runs periodically at a high frequency.
- Task Y (Medium Priority): Performs computations and may occasionally need access to the same shared resource used by Task X. Has a lower priority than Task X.
- Task Z (Low Priority): Executes less critical operations and requires access to the shared resource occasionally. Has the lowest priority among the three tasks.

Describe a scenario where priority inversion can occur in this system. Additionally, discuss potential solutions or protocols that can be implemented to mitigate priority inversion in this context.

c) What is the meaning of the term 'busy waiting' in the context of process synchronization? Explain whether busy waiting be avoided altogether. 1 + 4 (CO1) (PO1)