National University

Of Computer & Emerging Sciences, Peshawar Campus

Student Name: ____________________ Roll No: ________________

Program: **BSCS & BSSE** Examination: Final

Semester: **Fall-2023** Total Marks: 60 (Weightage 50%)

Time Allowed: 2 hours 30 min Date: Wed, Dec 13, 2023

Course: Operating Systems Lab Instructor: Muhammad Ahsan

Note:
- Attempt all questions.
- Don't rush, carefully understand the problem and then go for solution.
- Understanding the question is part of the Exam.
- An individual may be assigned with straight **ZERO** if the submitted assessed task is copied/cheated from another individual/internet/found using the internet.
- Create a folder named as Roll number, name, and section i.e (23P-1234_Name_5A), copy all the files (**Q1.c, Q2.c, Q3.sh, Q4.c**), in the folder and submit that folder.

Question # 1 [Marks 20]

Online streaming platforms need to maintain a repository of different formats of videos to cater for different user requirements. These video formats can vary in terms of video resolution, compression algorithm, etc. Each video format may have a different set of associated processes, each having a specific role. Distributed processing is the solution where each process feeds onto another to eventually generate the required format.

Considering this use case, let's propose a model of Distributed Video Processing system consisting of multiple processes which communicate using pipes in the distributed system to perform the complex video transcoding task.

Here are the processes involved:
1. PV1 (Input Process): This process is responsible for extracting video files from the repository. It generates file names and forwards them to PV2 for subsequent steps.
2. PV2 (Transcoding Process): The video file names are received from PV1 via a pipe. The responsibility of PV2 is to transcode the videos by changing their format while appending 'transcoded' at the end of each file name. Once done, PV2 transfers the transcoded files to PV3.
3. PV3 (Quality Control Process): The transcoded video files from PV2 are received by PV3 through a pipe. PV3 is intended to validate the resolution of transcoded videos, appending 'QC_Passed' at the end of each file name after successful validation. Afterward, it sends the validated files to PV4 for final delivery.

4. PV4 (Distribution Process): The validated files received from PV3 through a pipe are the responsibility of PV4. This process distributes the videos to different streaming platforms.

Use pipes for efficient communication between PV1, PV2, PV3, and PV4 in a video processing system. Each process focuses on its specific task, ensuring seamless and synchronized data exchange for effective management of large-scale video processing tasks.

Note: You can use fork() to create different processes and then implement your logic in each process accordingly.

Question # 2 [Marks 20]

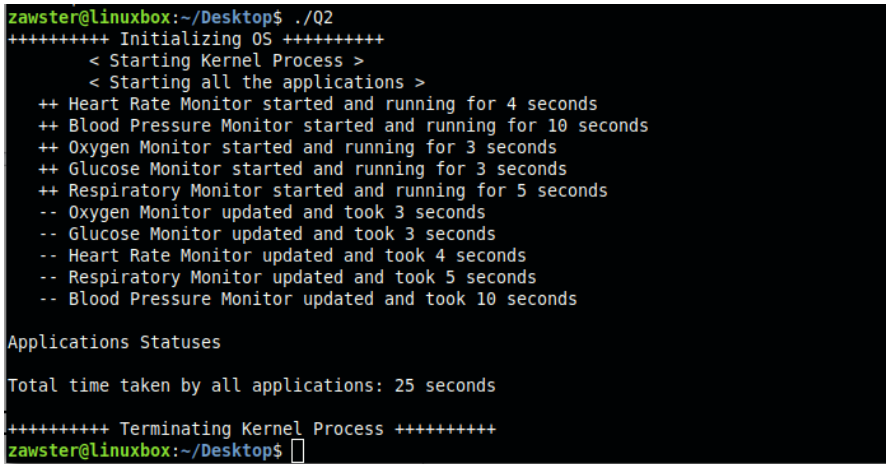
In the current health scenario, the importance of telehealth systems has become prominent. Therefore, we need to develop an embedded system for a telehealth monitoring system. This embedded system is accountable for managing various monitoring devices in the system, such as Heart Rate Monitors, Oxygen Monitors, Blood Pressure Monitors, Glucose Monitors, and Respiratory Monitors. Each device is considered as an application that runs on top of the operating system. The operating system should support concurrent execution of multiple device applications and provide intercommunication among them.
Your task is to write a code for the embedded system that initializes the system, creates threads for each device application, and manages their execution. Each device application must print an acknowledgment message at the beginning and update its status periodically. The embedded operating system should guarantee proper synchronization and termination of the device applications.

The list of device applications and their update intervals in seconds is as follows:

- ➢ Heart Rate Monitor (4 seconds)
- ➢ Oxygen Monitor (3 seconds)
- ➢ Blood Pressure Monitor (10 seconds)
- ➢ Glucose Monitor (3 seconds)
- ➢ Respiratory Monitor (5 seconds)

Your code should demonstrate the initialization of the embedded operating system, execution of device applications, and the termination of the system after all applications have completed.

Your Output is supposed to be like this:



```
zawster@linuxbox:~/Desktop$ ./Q2
++++++++++ Initializing OS ++++++++++
        < Starting Kernel Process >
        < Starting all the applications >
   ++ Heart Rate Monitor started and running for 4 seconds
   ++ Blood Pressure Monitor started and running for 10 seconds
   ++ Oxygen Monitor started and running for 3 seconds
   ++ Glucose Monitor started and running for 3 seconds
   ++ Respiratory Monitor started and running for 5 seconds
   -- Oxygen Monitor updated and took 3 seconds
   -- Glucose Monitor updated and took 3 seconds
   -- Heart Rate Monitor updated and took 4 seconds
   -- Respiratory Monitor updated and took 5 seconds
   -- Blood Pressure Monitor updated and took 10 seconds

Applications Statuses

Total time taken by all applications: 25 seconds

++++++++++ Terminating Kernel Process ++++++++++
zawster@linuxbox:~/Desktop$ 
```

After understanding the above problem, do the following:

1. Write a program in C using multithreading and parallelism.
2. Identify the data involved in the race condition.
3. Using a semaphore or mutex lock, fix the race condition.
4. You have to compare the total running time that was printed in your output with the actual execution time that you can roughly count on your fingertips while your output was printing.

Notes:

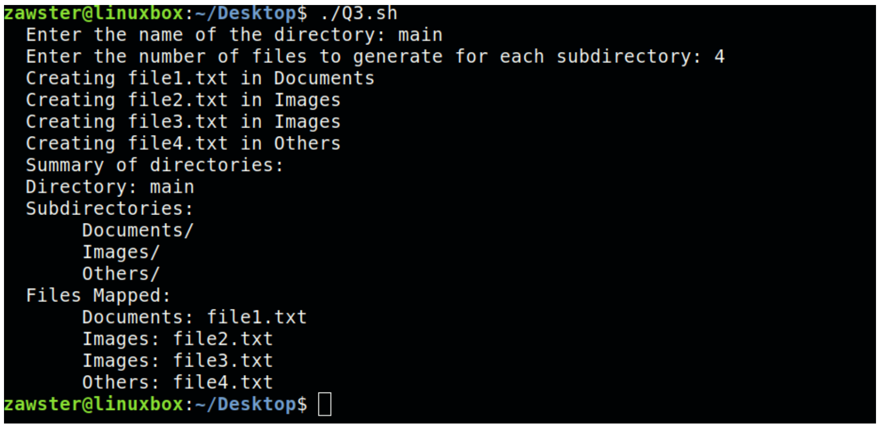
- Kernel is also an application that will start first and close after execution of all other applications.
- The system cannot be shut down until all applications are closed.
- You can define the runningTime variable globally.

Question # 3 [Marks 10]

Create a Shell Script to automate a file management task. The script should perform the following actions:

1. Ask the user to enter the name of a directory.
2. Check if the directory exists. If not, create the directory.
3. Inside the directory, create three subdirectories named “Images”, “Documents”, and “Others”.
4. Ask from the user to enter the number of files they want to generate for each subdirectory.
5. Using loops, generate empty files with random names in each subdirectory based on the user's input.
6. For each file, randomly assign (of your choice) it to one of the subdirectories (that you just created) using conditionals.
7. Display a summary of the created directories, subdirectories, and files.

Ensure that your script includes proper comments to explain each section. Your Output is supposed to be like this:

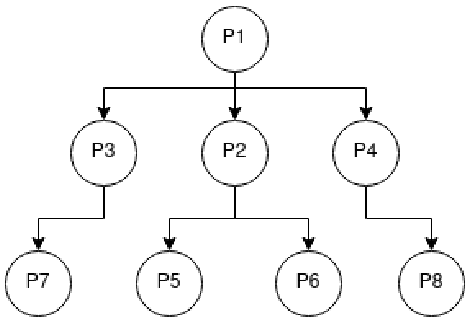


```
zawster@linuxbox:~/Desktop$ ./Q3.sh
  Enter the name of the directory: main
  Enter the number of files to generate for each subdirectory: 4
  Creating file1.txt in Documents
  Creating file2.txt in Images
  Creating file3.txt in Images
  Creating file4.txt in Others
  Summary of directories:
  Directory: main
  Subdirectories:
       Documents/
       Images/
       Others/
  Files Mapped:
       Documents: file1.txt
       Images: file2.txt
       Images: file3.txt
       Others: file4.txt
zawster@linuxbox:~/Desktop$ 
```

Bonus (Optional): Implement error handling to check if the user provides a valid directory name and a valid number of files to generate. If invalid inputs are detected, display appropriate error messages and exit the script with a goodbye message.

Question # 4 [Marks 10]

Implement the following process tree using fork system call.



NOTE: The process tree should be constructed as follows:

1. P1 is the parent process, and it should create the rest of the processes as mentioned in the tree.
2. There should be a total of 8 processes, you can skip the child process of fork() at some points in the tree.

NOTE: There is Zero tolerance to plagiarism. If any solution to any question is caught in any sort of plagiarism, ZERO marks will be assigned to that question straight forward.

GOOD LUCK ☺