National University
of computer and emerging sciences

Course Name:	**Operating System**	**Course Code**	**CS2006**
Degree Program:	**BSSE**	**Semester:**	**Fall 2024**
Exam Duration:	**90 Minutes + 10 minutes for submission**	**Total Marks:**	**30**
Paper Date:	**14-October-2024**	**Weightage:**	**30 %**
Section:	**5C**	**Exam Type:**	**MidTerm**

Instructions:

- You can use Linux man pages for help.
- Understanding the question statement is also part of the exam, so do not ask for clarification.
- You must ensure proper code submission following the file naming instructions.
- Empty Submission means ZERO marks. You are not allowed to recover your files after the exam's commencement.
- Your submission will contain your code and output screenshots.
- Submissions without roll_number (or question no) on it will be evaluated against ZERO.
- Submission location: Cactus.
- Use C language to write all the source code files.
- Students will receive ZERO marks if the answers are plagiarized.
- Use of ANY helping material/code, cell phones, INTERNET, and flash drives are strictly prohibited.
- **Low Space Error:** If you are facing this error, It is advised to delete your current virtual machine. Find C:/NS2.zip file. Open the virtual machine in vmware after unzipping it. While creating the virtual machine uplift the allocated RAM to alteast 35 GB.

Question:01 **Marks:20**

Multi-process communication system for the stock market

You are required to design a multi-process communication system for the XYZ company that processes real-time stock market data.

The system should consist of three distinct processes.

Process A:

A **data producer** that continuously fetches stock prices and writes them to a Named Pipe.

Process B:

A **data logger** that reads stock prices from the Named Pipe and writes the prices to a log file for future analysis.

Process C:

A **real-time analysis tool** that also reads from the Named Pipe but displays processed stock information (e.g., moving averages) on terminal.

The goal is to build an efficient inter-process communication pipeline where:

- **Process A** writes stock data to a Named Pipe, and **Process B** and **Process C** both read the data from the pipe simultaneously.
- To achieve this, **Process B** and **Process C** must each duplicate the Named Pipe's file descriptor using dup() to either redirect the pipe to their log file (in the case of Process B) or to standard output (in the case of Process C).

However, since only one reader can receive data from a pipe at a time, you are required to implement two separate Named Pipes: one for communication between Process A and Process B, and another for Process A and Process C.

You are required to perform these tasks:

I. **Process A** continuously writes stock price data in the format "Stock: XYZ, Price: 123.45" to both Named Pipes.
II. **Process B** reads data from its Named Pipe, uses dup() to redirect its output to a log file (log.txt), and writes the stock data to the log file.
III. **Process C** reads from its Named Pipe, uses dup() to redirect output to standard output, and prints real-time stock data on the terminal.

→ Make three separate files and name them "producer.c","logger.c" and "realtimedisplay.c"
→ You can run the processes on three different terminals or you can use '&' operator to run them on one terminal.
→ **All processes** should handle file descriptors properly, ensuring they are closed once communication is complete.

Question:02 **Marks:10**

Data Processing Application

Imagine you're designing a simple data processing application that simulates part of a network packet inspection system. The system consists of two main tasks:

1. **Packet Generator (Child Process 1)**: The first child process generates a series of packet identifiers (for simplicity, numbers from 1 to 100) and sends this data to the second process for filtering.
2. **Packet Filter (Child Process 2)**: The second child process receives the packet identifiers from the first process via an **ordinary pip**e and filters out the packet numbers that are even, printing only the odd packet numbers.
3. **Parent Process**: The parent process supervises both child processes by creating the pipe, forking the children, and waiting for them to complete. The parent process does not directly handle the data.
4. **Your task**: Implement this scenario using ordinary pipes (unnamed pipes). Ensure proper communication between the processes. The packet generator should send numbers (packets), and the packet filter should only display the odd numbers (packets) on the terminal.

Output:

Packet 1 is odd
Packet 3 is odd
Packet 5 is odd
.....
Packet 99 is odd