National University of Computer and Emerging Sciences

Laboratory Manual # 07
Operating Systems

Course Instructor	Mubashar Hussain
Lab Instructor	Muhammad Hashir Mohsineen
Section	BCS-4E
Date	25-March-2025
Semester	Spring 25

Instructions:

- Submit a world/LibreOffice file containing screenshots of terminal commands/ Output
- Submit your .c (Code files)
- In case of any explanation you can add a multiline comment.

Objectives:

- Shared Memory

Reading material:
https://docs.google.com/document/d/1OvZZ-MAkXwX8xKyqZh4ay4uPhf_Q2yReEAwARfOOgAA/edit?usp=sharing

1. Exercise: [10]

Create 2 processes, client and server:
Server Process:

- Creates a shared memory segment using a unique key.
- Waits for the client process to write data into the shared memory.
- Reads the data from shared memory, calculates the sum and average, and displays the results.
- Deletes the shared memory segment after reading and processing the data.

Client Process:

- Attaches to the shared memory segment created by the server.
- Reads data from a file (e.g., number.txt file name passed by command line arguments) and writes it to the shared memory segment.
- Detaches from the shared memory after writing the data.

2. Exercise: [10]

Write a program that uses shared memory to aggregate data from multiple processes. The goal is to calculate the sum of an array of integers, where each process computes a partial sum of a subset of the array and stores it in a shared memory segment. The main process will then read the partial sums from the shared memory and compute the final result.

details:

1. Define a shared memory segment large enough to store:
    - An array of integers (size N).

    - An array of partial sums (size M, where M is the number of processes).
    - A final sum variable.
2. The main process should:
    - Create the shared memory segment.
    - Initialize the array of integers with random values (using rand()).
    - Fork M child processes.
3. Each child process should:
    - Compute the sum of a subset of the array (divide the array into M equal parts).
    - Write its partial sum into the shared memory segment at the appropriate index.
    - Exit after writing the partial sum.
4. The main process should:
    - Wait for all child processes to complete.
    - Read the partial sums from the shared memory and compute the final sum.
    - Print the final sum.
    - Detach and delete the shared memory segment.

🙂