National University of Computer and Emerging Sciences


# Operating Systems Lab

# Lab Manual 7

**Muhammad Hassan Raza**

**Fall 2024**

**Department of Software Engineering**
**FAST-NU, Lahore, Pakistan**

## Question 1: Complex Number Array Processing Using Shared Memory (20 marks)

You are required to write a program that takes an array of complex numbers as input and performs a series of operations using shared memory. The program should handle the input array, process it using multiple child processes, and return the results.

Input Format:
- The first argument is the size of the array `N`.
- The next `2*N` arguments represent the real and imaginary parts of the complex numbers. For each complex number, the real part comes first, followed by the imaginary part.

For example, the input format for 3 complex numbers would be:
./program 3 1 2 3 4 5 6
This represents the complex numbers:
1 + 2i, 3 + 4i, 5 + 6i

Tasks:

1. Shared Memory Setup:
   - Use shared memory to store the array of complex numbers.
   - Use separate shared memory blocks to store the intermediate and final results.

2. Fork Processes:
   - The parent process will fork three child processes (Child 1, Child 2, Child 3), each performing a different operation on the array.

   Child Process 1 (Complex Conjugate):
   - This process will compute the complex conjugate of each number in the array (i.e., invert the sign of the imaginary part).
   - Store the result in shared memory block 1.

   Child Process 2 (Magnitude Calculation):
   - This process will calculate the magnitude of each complex number. The magnitude of a complex number `a + bi` is `sqrt(a^2 + b^2)`.
   - Store the result in shared memory block 2.

   Child Process 3 (Sum of Complex Numbers):
   - This process will compute the sum of all complex numbers in the array.

- Store the result in shared memory block 3.

3. Parent Process:
   - The parent process waits for all child processes to complete.
   - It then reads the results from the shared memory:
     - Conjugates,
     - Magnitudes,
     - Sum of the complex numbers.
   - The parent process will print all results.

4. Expected Output:
   - Print the complex conjugates.
   - Print the magnitudes of the complex numbers.
   - Print the sum of the complex numbers.

## Example:

Example Input:
./program 2 1 2 3 4

Example Output:
Conjugates: (1 - 2i), (3 - 4i) Magnitudes: 2.236, 5.0 Sum: (4 + 6i)

Constraints:
- Ensure proper synchronization when accessing shared memory between processes.
- Handle arrays of up to 100 complex numbers.

## Question 2: Multi-Stage File Processing Using Named Pipes (20 marks)

You are tasked with creating a multi-stage file processing system using named pipes (`FIFO`). The program will read an array of integers from a file and perform multiple transformations across different stages.

Stages:

1. Input Format:
   - The first argument is the path to a text file that contains space-separated integers.
   - The program should read these integers and pass them through several stages of processing.

2. Named Pipes Setup:
   - Create three named pipes (`fifo1`, `fifo2`, `fifo3`) for inter-process communication.

3. Fork Processes:
   - The program must fork three child processes that communicate using the named pipes to complete the following tasks:

   Stage 1 (Child Process 1):
   - Read the integers from the file and compute the square of each number.
   - Write the squared numbers to `fifo1`.

   Stage 2 (Child Process 2):
   - Read the squared numbers from `fifo1`.
   - Compute the running average of the squared numbers and write the averages to `fifo2`.

   Stage 3 (Child Process 3):
   - Read the running averages from `fifo2`.
   - Identify and count how many of the running averages are greater than a user-specified threshold (provided as the second command-line argument).
   - Write the final count to `fifo3`.

4. Parent Process:
   - The parent process will read the final count from `fifo3` and display the result.

Expected Output:
- Print the final count of running averages greater than the threshold.

## Example:

- Input file (`input.txt`):
2 4 6 8 10

- Threshold: 20

- Expected Output:
2 running averages are greater than 20.

Constraints:
- Handle input files with up to 100 integers.
- Ensure synchronization when writing to and reading from the named pipes.