LAB 11
TASK 17 – Shared Memory
TASK 18 - Message Queues

John Dempsey
COMP-232: Programming Languages
California State University Channel Islands
https://comp232.com

Task 17. Shared Memory

Shared memory allows programs to create a memory area which can be read and written by one or more programs while the programs are running.

In this assignment, you will write four shared memory programs. These programs will do the following:

Program 1 will create the shared memory segment then exit.

Program 2 will write a message to the shared memory segment, e.g., writeshm “Hello World”.

Program 3 will read the message, if any, stored on the memory segment.

Program 4 will remove the shared memory.

You can use ChatGPT to write the above programs. You just need to get them to work.

Task 18. Message Queues

Message queues set up a queue to allow running programs to send messages from one process to another.

For example, claims can be sent to a data center, received by a network process, and placed on an application message queue for processing. Processing can be to pay or deny a claim or determine if a recipient is eligible for a program. After the claim has been processed, the application process places the response on a message queue to the network process so the response can be sent back to the network vendor.

In this assignment, you will write six programs, which will do the following:

1. Program create_queue.c will create a message queue.
2. Program a100.c will write a message into the message queue with a priority of 100.
3. Program a200.c will write a message into the same message queue with a priority of 200.
4. Program b100.c will read a message from the message queue with a priority of 100,
5. Program b200.c will read a message from the message queue with a priority of 200.
6. Program rm_queue.c will delete the message queue.

You can use ChatGPT to create the six programs above.

You'll need to test for the following:

1. Running b100 before running create_queue.
2. Running rm_queue before running b100.
3. Running b100 before running a100.
4. Running b200 before running a200.
5. Running b100 and then running b200.
6. Running b200 and then running b100.
7. Running a100 four times in a row, then running a200, then run a100 once, and a200 twice.