ECE 2420 Programming Exercise #4

(Processes and IPC)

# Overview

This programming exercise utilizes various multi-processing and inter-process communication mechanisms. The purpose of this code is to emulate a distributes system in which one process composes problems to be solved. These problems are passed to the second process which solves the problem. This second process then passes the result to a third process which consumes the result.

# Produce and Consumer Architecture

This PEX consists of two sets of problems. The first uses message queues, and the second uses shared memory. Each set is broken down into two phases. The first phase of each problem consists of only two processes where the producing process and the consuming process are the same. The second phase of each problem separates the producing and consuming processes. The architecture of each are shown below:

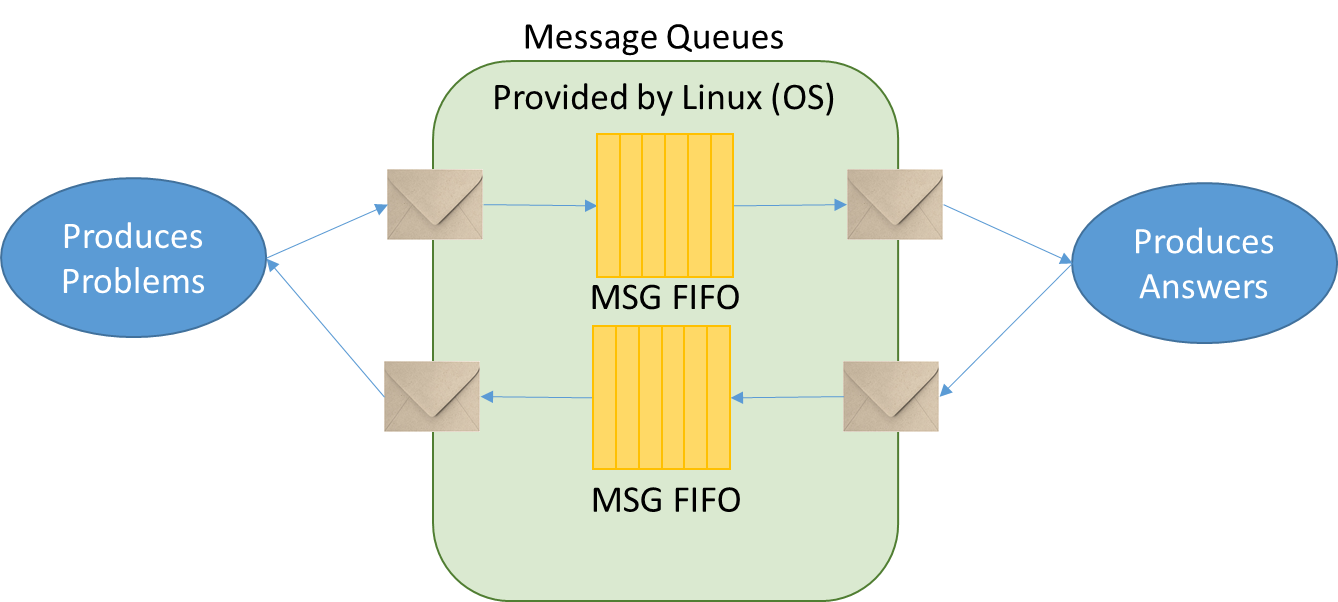


Figure 1. Message Queue Phase 1

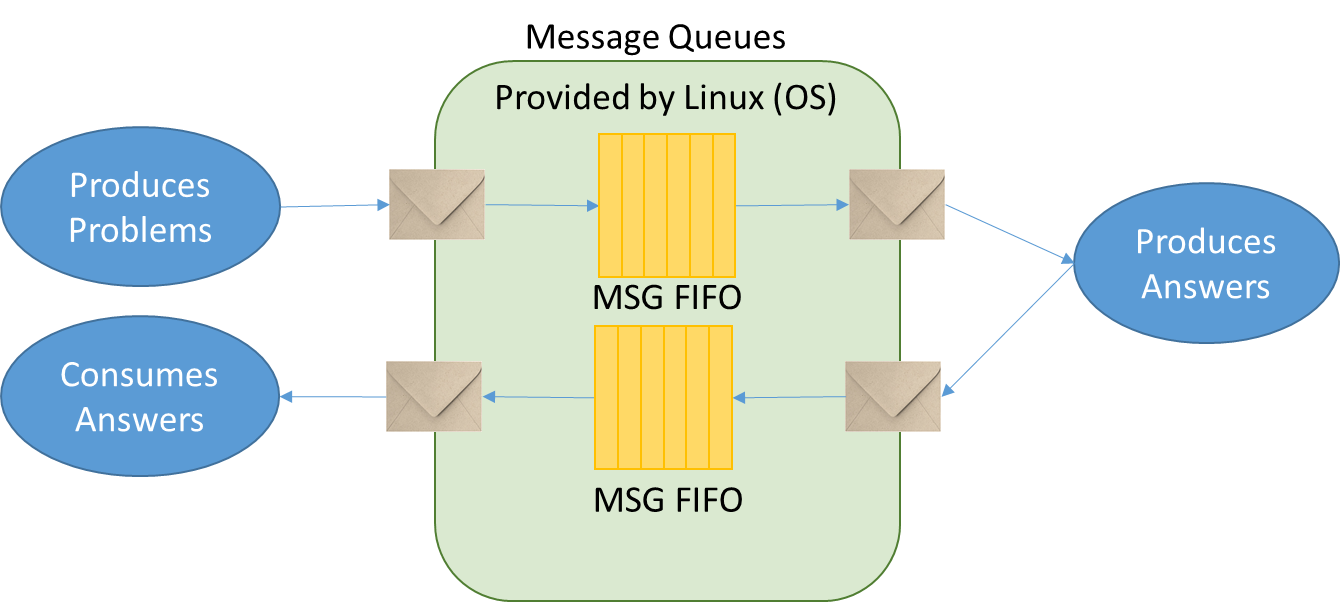


Figure 2. Message Queue Phase 2

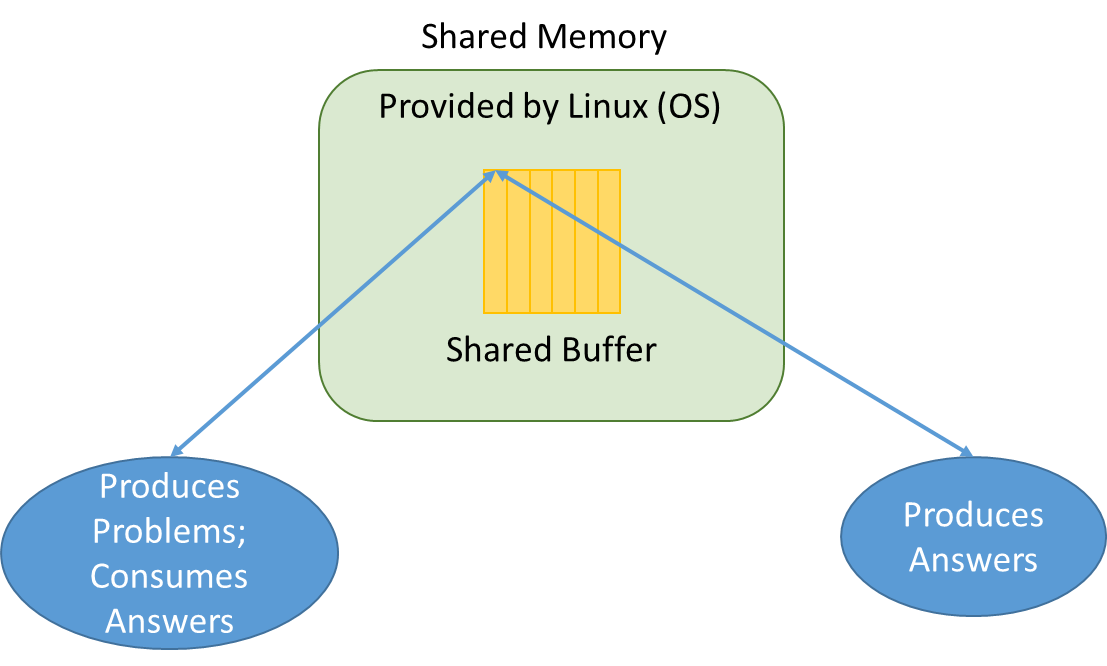


Figure 3. Shared Memory Phase 1

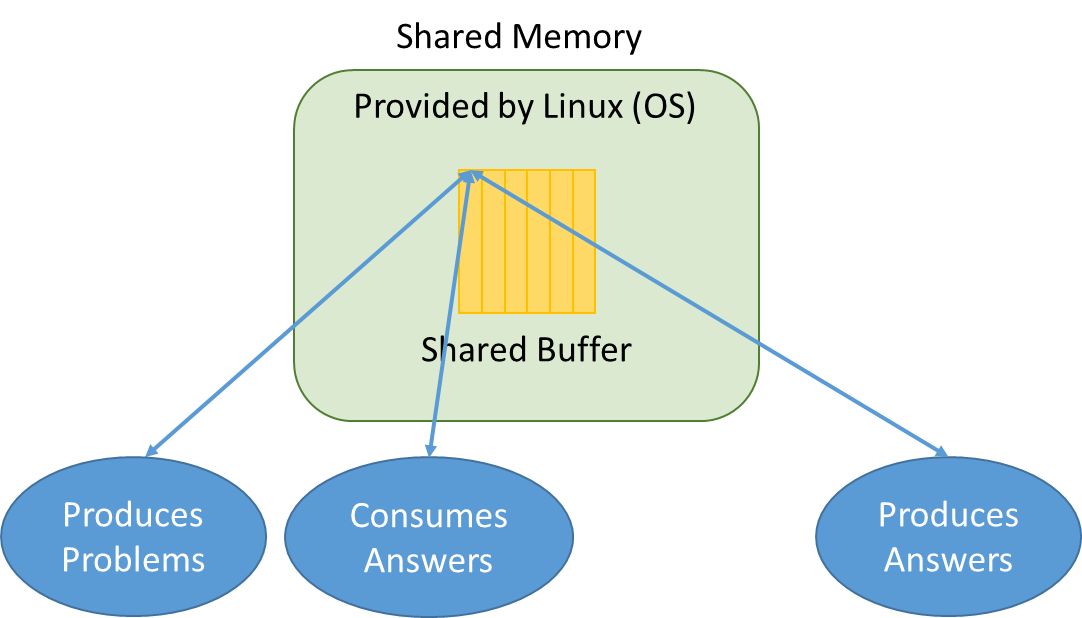


Figure 4. Share Memory Phase 2

Regardless of which architecture or phase is being developed, a producing process will randomly generate a trivial math problem. This problem is stored in the C structure called “Problem” in the provided header. The problem is then placed in a message queue or shared memory where another process will execute the problem and produce an “Answer” object. The answer is then placed in another location where either the originating process or a third process can consume the result.

# Programming Concepts

This exercise covers many programming concepts including processes, named message queues, named semaphores, shared memory, random number generation, and producer consumer architectures.

# System Requirements

The design must use the provided header verbatim. All inter process communication primitives should be named as specified in the provided header. This will allow automated testing of the design you produce.

Here are addition specific requirements that shall be implemented/followed:

1. The service process shall create all needed queues, memory spaces, semaphores, etc.
2. The service process shall initialize all IPC primitives to the proper initial state
3. The producing process connects to IPC primitives created by the service
4. The producing process creates Problem objects and passes them to the service process
   1. Message queues are used for the first two problems
   2. Shared memory is used for the second two phases
5. The service process receives the Problem objects, executes the operation, and produces an Answer object
6. In the first phase of each problem, the Answer is passed back to the producing process.
7. In the second phase, the Answer is passed back to the consumer process
8. In the first phase, the producing process will wait until it receives and processes an Answer before sending a new Problem
9. In the second phase, the producing process will create and queue Problems as quickly as the service can accept them
10. The service shall buffer ARRAY\_SIZE Problems as specified in the provided header
11. For each problem and phase, you should provide the following analysis:
    1. Time for executing 1 iteration of ARRAY\_SIZE
    2. Time for executing 10 iterations of ARRAY\_SIZE
    3. Time for executing 100 iterations of ARRAY\_SIZE
    4. Time for executing 1000 iterations of ARRAY\_SIZE
    5. In prose, compare and contrast the performance of each and provide an explanation for your observed results

# Turn-in Procedures

Turn in all source code and required analysis via canvas by 11:59p.m. on <DUE DATE>

Grading Rubric

(ECE 2420 PEX4)

|  |  |  |
| --- | --- | --- |
| Requirement / Criteria | Available Points | Student’s Score |
| Msg queue producer places Problems in queue | 10 |  |
| Msg queue service retrieves, executes and produces answers | 10 |  |
| In phase 1, Msg queue producer receives responses | 10 |  |
| In phase 2, Msg queue consumer receives messages and ARRAY\_SIZE problems are allowed “in flight” | 10 |  |
| Shared mem producer places Problems in queue | 10 |  |
| Shared mem service retrieves, executes and produces answers | 10 |  |
| In phase 1, Shared mem producer receives responses | 10 |  |
| In phase 2, Shared mem consumer receives messages and ARRAY\_SIZE problems are allowed “in flight” | 10 |  |
| Written analysis is complete and rational | 20 |  |
| Total | **100** |  |