Group-Activity-1

**Group Members:**

Abhishek Srivastava - 19BCE10071  
Samyak Jain - 19BCE10066  
Vishesh bhadauria - 19BCE10304

**Title:** Study and Simulate IPC in Posix environment.

**Procedure:**

*Theoretical Explanation*

There are two processes: Producer and Consumer. The producer produces some items and the Consumer consumes that item. The two processes share a common space or memory location known as a buffer where the item produced by the Producer is stored and from which the Consumer consumes the item if needed.

There are two versions of this problem: the first one is known as the unbounded buffer problem in which the Producer can keep on producing items and there is no limit on the size of the buffer, the second one is known as the bounded buffer problem in which the Producer can produce up to a certain number of items before it starts waiting for Consumer to consume it.

We will discuss the bounded buffer problem. First, the Producer and the Consumer will share some common memory, then the producer will start producing items. If the total produced item is equal to the size of the buffer, the producer will wait to get it consumed by the Consumer. Similarly, the consumer will first check for the availability of the item. If no item is available, the Consumer will wait for the Producer to produce it. If there are items available, Consumer will consume them.

**Simulation Example:**

*Producer process illustrating POSIX shared-memory API.*

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <fcntl.h>**

**#include <sys/shm.h>**

**#include <sys/stat.h>**

**int main()**

**{**

**const int SIZE = 4096;**

**const char\* name = "OS";**

**const char\* message\_0 = "Hello";**

**const char\* message\_1 = "World!";**

**int shm\_fd;**

**void\* ptr;**

**shm\_fd = shm\_open(name, O\_CREAT | O\_RDWR, 0666);**

**ftruncate(shm\_fd, SIZE);**

**ptr = mmap(0, SIZE, PROT\_WRITE, MAP\_SHARED, shm\_fd, 0);**

**sprintf(ptr, "%s", message\_0);**

**ptr += strlen(message\_0);**

**sprintf(ptr, "%s", message1);**

**ptr += strlen(message\_1);**

**return 0;**

**}**

*Consumer process illustrating POSIX shared-memory API.*

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <fcntl.h>**

**#include <sys/shm.h>**

**#include <sys/stat.h>**

**int main()**

**{**

**const int SIZE = 4096;**

**const char\* name = "OS";**

**int shm\_fd;**

**void\* ptr;**

**shm\_fd = shm\_open(name, O\_RDONLY, 0666);**

**ptr = mmap(0, SIZE, PROT\_READ, MAP\_SHARED, shm\_fd, 0);**

**printf("%s", (char\*)ptr);**

**shm\_unlink(name);**

**return 0;**

**}**

Reference:

POSIX API for shared memory, Message passing in the Mac operating system. and Windows XP shared memory as a mechanism for providing certain types of message passing concepts and implementation as given in your textbook.