**Operating System**

**Lab Report 9**

**Hafiz Ahmad**

**19l-1316**

**Section-6A2**

Semaphores using Shared Memory

**INTRODUCTION:**

A semaphore lock on the shared memory buffer reference allows processes accessing the shared memory to prevent a daemon from writing to the memory segment currently being accessed. A situation that several tasks access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access take place. Example: Suppose that the value of the variable counter = 5. Process 1 and process 2 execute the statements “counter++” and “counter--” concurrently

**OBJECTIVES:**

• Learn the concept of semaphores

• Understand the use of Shared Memory

• Learn to use semaphores for synchronization in Inter Process Communication

**Application:**

Semaphores protect access to resources such as shared memory. Before writing to a shared memory region, a process can lock the semaphore to prevent another process from accessing the region until the write operation is completed. A semaphore lock on the shared memory buffer reference allows processes accessing the shared memory to prevent a daemon from writing to the memory segment currently being accessed. Two processes can both attach to a shared memory segment. A shared memory segment wouldn't be much use if that were not true, as that is the basic idea behind a shared memory segment - that's why it's one of several forms of IPC (inter-Process communication).

**Issues:**

No issue found regarding this lab.

**Conclusion:**

In this lab we learn the concept of semaphores. We also Understand the use of Shared Memory and the use of semaphores for synchronization in Inter Process Communication. Shared memory is memory that may be simultaneously accessed by multiple programs with an intent **to provide communication among them or avoid redundant copies**. Shared memory is an efficient means of passing data between programs.