YAŞAR UNIVERSITY

FACULTY OF ENGINEERING

COMP 3323 Operating Systems Laboratory Sheet

LAB 6: Producer-Consumer Problem (Semaphores): Using semaphore, we will solve the problem in a limited buffer situation.

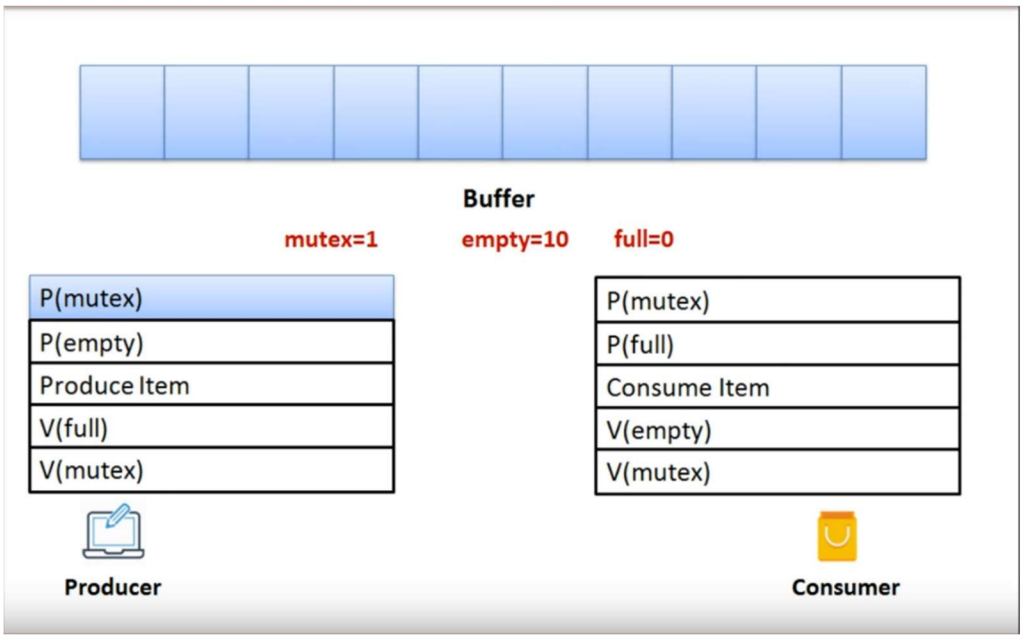
- ***Producer and consumer are created as processes.***

- There are **3 important restrictions here:**

1. Buffer can only access one process at a time (mutual exclusion). For this purpose, a mutual semaphore named mutex will be used.

2. If Buffer is empty, the consumer waits for the manufacturer to enter data in Buffer. The producer and consumer will use the empty semaphore for synchronization.

3. If Buffer is full, the manufacturer waits for the consumer to receive data from Buffer. The producer and consumer will use the full semaphore named for synchronization.



Producer - Consumer Problem Pseudocode

```
# define N 100       //Number of elements in Buffer
# define TRUE 1
typedef int semaphore; //Semaphores are defined as an int.
semaphore mutex= 1;    //mutual exclusion of critical part
semaphore empty = N;   //The number of empty space in the buffer

semaphore full = 0;   //The number of filled places in Buffer
producer() {
    int item;
    while (TRUE) {
        produce_item(&item); //Generating data to be placed in the buffer.
        wait(empty);     //Wait if the buffer is full, or reduce the number of empty spaces by 1.
        wait(mutex);       //Get permission to enter the critical section.
        enter_item(item); //Enter the data in Buffer (Critical Section).
        signal(mutex);     //Indicate that it exits the Critical Section.
        signal(full); //If there are any consumers waiting, wake up,
                      //or increase the number of full places  in Buffer by 1.
    }
}
consumer(){
     int item;
     while (TRUE) {
         wait(full);     //Wait if the buffer is empty, or reduce the number of full places by 1
         wait(mutex);       //Get permission to enter the critical section
         remove_item(&item); //Get data from Buffer (Critical Part)
         signal(mutex);              //Indicate that it exits the Critical Section
         signal(empty); //If there is a manufacturer waiting, wake up,
                        //  or increase the number of empty  spaces in Buffer by 1.
         consume_item(item); //Use data from Buffer.
         }
     }
```