

Ex.No: 9	PRODUCER CONSUMER PROBLEM USING SEMAPHORES
---------------------------	---

AIM:
To write a C-program to implement the producer–consumer problem using semaphores.

ALGORITHM:

- Step1: Start the program.
- Step2: Declare the required variables.
- Step3: Initialize the buffer size and get maximum item you want to produce.
- Step4: Get the option, which you want to do either producer, consumer or exit from the operation.
- Step5: If you select the producer, check the buffer size, if it is full the producer should not produce the item or otherwise produce the item and increase the value buffer size.
- Step6: If you select the consumer, check the buffer size if it is empty the consumer should not consume the item or otherwise consume the item and decrease the value of buffer size.
- Step7: If you select exit come out of the program.
- Step8: Stop the program.

PROGRAM:

```
#include<stdio.h>
```

```
int mutex=1,full=0,empty=3,x=0; // Number of empty slots as  
size of buffer //
```

```
main()  
{  
int n;  
void producer();  
void consumer();  
int wait(int);  
int signal(int);  
printf("\n1.PRODUCER\n2.CONSUMER\n3.EXIT\n");  
while(1)  
{  
    printf("\nENTER YOUR CHOICE\n");  
    scanf("%d",&n);  
    switch(n)  
    {  
case 1:  
        if((mutex==1)&&(empty!=0))  
            producer();  
        else  
            printf("BUFFER IS FULL);  
        break;  
  
case 2:  
        if((mutex==1)&&(full!=0))  
            consumer();  
        else  
            printf("BUFFER IS EMPTY");  
        break;
```

```
case 3:
    exit(0);
break;
}
}
}
```

```
int wait(int s)
{
return(--s);
}
```

```
int signal(int s)
{
return(++s);
}
```

```
void producer()
{
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
    printf("\nproducer produces the item%d",x);
    mutex=signal(mutex);
}
```

```
void consumer()
{
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\n consumer consumes item%d",x);
x--;
mutex=signal(mutex);
}
```