**Experiment No: 9**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS-A-1 |
| **Link to Code:** https://github.com/NamitKumar16/OS |
| **Date:** 20th October 2021 |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:** Write a program to implement reader/writer problem using semaphore |
| **Program Outcome**  The students will understand the reader/writer problem using semaphore |
| **Problem Statement:**  Write a program to implement reader/writer problem using semaphore |
| **Background Study:** There is a shared resource which should be accessed by multiple processes. There are two types of processes in this context. They are reader and writer. Any number of readers can read from the shared resource simultaneously, but only one writer can write to the shared resource. When a writer is writing data to the resource, no other process can access the resource. A writer cannot write to the resource if there are non-zero number of readers accessing the resource at that time. |
| **Question Bank:**   1. An un-interruptible unit is known as \_\_\_\_\_\_\_\_\_\_\_\_ a) single b) atomic c) static d) none of the mentioned 2. TestAndSet instruction is executed \_\_\_\_\_\_\_\_\_\_\_\_ a) after a particular process b) periodically c) atomically d) none of the mentioned 3. Semaphore is a/an \_\_\_\_\_\_\_ to solve the critical section problem. a) hardware for a system b) special program for a system c) integer variable d) none of the mentioned 4. What are the two atomic operations permissible on semaphores? a) wait b) stop c) hold d) none of the mentioned 5. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called \_\_\_\_\_\_\_\_ a) dynamic condition b) race condition c) essential condition d) critical condition |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

pthread\_mutex\_t wr,mutex;

int a = 10,readcount=0;

void \* reader(void \*arg)

{

long int num;

num=(long int) arg;

pthread\_mutex\_lock(&mutex);

readcount++;

pthread\_mutex\_unlock(&mutex);

if(readcount==1)

{

pthread\_mutex\_lock(&wr);

}

printf("\nReader %ld is in critical section",num);

printf("\nReader %ld is reading data %d",num,a);

sleep(1);

pthread\_mutex\_lock(&mutex);

readcount--;

pthread\_mutex\_unlock(&mutex);

if(readcount==0)

{

pthread\_mutex\_unlock(&wr);

}

printf("\nReader %ld left criticial section",num);

}

void \* writer(void \*arg)

{

long int num;

num=(long int) arg;

pthread\_mutex\_lock(&wr);

printf("\nWriter %ld is in critical section",num);

printf("\nWriter %ld have written data as %d",num,++a);

sleep(1);

pthread\_mutex\_unlock(&wr);

printf("\nWriter %ld left critical section",num);

}

int main()

{

pthread\_t r[10],w[10];

long int i,j;

int nor,now;

pthread\_mutex\_init(&wr,NULL);

pthread\_mutex\_init(&mutex,NULL);

printf("Enter number of readers and writers\t");

scanf("%d %d",&nor,&now);

for(i=0;i<nor;i++)

{

pthread\_create(&r[i],NULL,reader,(void \*)i);

}

for(j=0;j<now;j++)

{

pthread\_create(&w[j],NULL,writer,(void \*)j);

}

for(i=0;i<nor;i++)

{

pthread\_join(r[i],NULL);

}

for(j=0;j<now;j++)

{

pthread\_join(w[j],NULL);

}

return 0;

}

Text

Description automatically generated