OS LAB 7

NAME: Aditya Anand

ROLL NO.: 20124009

BRANCH: IT

S NO.	TITLE	DATE OF IMPLEMENTATION	REMARKS
1	Implementation of solution of Reader-Writer Problem	02-03-2022	
2	Implementation of solution of Sleeping Barber Problem	02-03-2022	

IMPLEMENTATION OF SOLUTION OF READER-WRITER PROBLEM

Consider a situation where we have a file shared between many people.
If one of the people tries editing the file, no other person should be reading or writing at the same time, otherwise changes will not be visible to him/her. However if some person is reading the file, then others may read it at the same time.

Precisely in OS we call this situation as the **readers-writers problem**

Problem parameters:

- One set of data is shared among a number of processes
- Once a writer is ready, it performs its write. Only one writer may write at a time
- If a process is writing, no other process can read it
- If at least one reader is reading, no other process can write
- Readers may not write and only read

Solution: We create two binary semaphores, mutex- to synchronise the increment and decrement of the reader_count variable and db- to synchronise and restrict access to readers and writers based on the problem statement conditions. We also have a struct database to represent an actual database of the real world.

CODE:

```
// C++ implementation of solution of Reader-Writer Problem using Binary Semaphore (Process
Synchronisation)
#include<bits/stdc++.h>
using namespace :: std;

struct Semaphore{
    bool s;

    Semaphore(){
        this->s=1;
    }

    void down(){
        if(this->s==0){
            cout<<"RESTRICTED!\n";
            return;
        }
        this->s=this->s-1;
    }

    void up(){
        this->s=this->s+1;
    }
};

// structure to represent the database
struct database{
    int value;
};
```

```
int rc=0;  // Reader Count

void Reader(Semaphore &mutex, Semaphore &db, database &d){
    mutex.down();
    rc++;
    if(rc==1){
        db.down();
    }
    mutex.up();

    // ---------------------------------------------------------------------
    cout<<"Reader "<<rc<<" is reading the database\n";
    cout<<"Value stored in the database = "<<d.value<<"\n\n";
    // ---------------------------------------------------------------------

    mutex.down();
    rc--;
    if(rc==0){
        db.up();
    }
    mutex.up();
}

void Writer(Semaphore &mutex, Semaphore &db, database &d, int val){
    db.down();

    // ---------------------------------------------------------------------
    cout<<"Writer is writing in the database\n";
    d.value=val;
    cout<<"Value updated!\n\n";
    // ---------------------------------------------------------------------

    db.up();
}

int main(){
    cout<<"SOLUTION TO READER-WRITER PROBLEM USING SEMAPHORE C++ IMPLEMENTATION\n";
    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";

    Semaphore mutex;   // Binary semaphore to synchronise incrementing of reader count
    Semaphore db;      // Binary semaphore to synchronise the access to a database

    database d;
    d.value = 9;      // Suppose the database stores the value 9 initially

    while(1){
        int n=0;
        cout<<"Enter \n1 for reader\n2 for writer\n3 to exit\n";
        cin>>n;
        if(n==1){
            Reader(mutex, db, d);
```

```
        }
        else if(n==2){
            cout<<"Enter the value you want to write: ";
            int val=0;
            cin>>val;
            Writer(mutex, db, d, val);
        }
        else break;
    }

    return 0;
}
```

RESULT:

```
PS C:\Users\beadi\Desktop\OS LAB\Assignment 7> cd "c:\Users\beadi\Desktop\OS LAB\Assignment 7\" ;
em }
SOLUTION TO READER-WRITER PROBLEM USING SEMAPHORE C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009       Branch: IT


Enter
1 for reader
2 for writer
3 to exit
1
Reader 1 is reading the database
Value stored in the database = 9

Enter
1 for reader
2 for writer
3 to exit
2
Enter the value you want to write: 5
Writer is writing in the database
Value updated!

Enter
1 for reader
2 for writer
3 to exit
1
Reader 1 is reading the database
Value stored in the database = 5

Enter
1 for reader
2 for writer
3 to exit
3
```

IMPLEMENTATION OF SOLUTION OF SLEEPING BARBER PROBLEM

Problem : The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barber chair, and n chairs for waiting for customers if there are any to sit on the chair.

- If there is no customer, then the barber sleeps in his own chair.
- When a customer arrives, he has to wake up the barber.
- If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.

Solution: Here we have used 2 binary semaphores, barber- to synchronise sleeping/awake state of the barber and cut- to synchronise the process of cutting of hair. We also have used a counting semaphore freeChairs- to synchronise the number of customers waiting in the waiting room.

In the following code, we have made an assumption that the customers arrive in groups and the next group arrives only when the previous group has been dealt with.

CODE:

```
// C++ implementation of solution of Sleeping Barber Problem using Semaphore (Process Synchronisation)
#include<bits/stdc++.h>
using namespace :: std;

struct binarySemaphore{
    bool s;
};

struct countingSemaphore{
    int s;
};

void customer(binarySemaphore &barber, binarySemaphore &cut, int Customers){
    int id=1;
    while(id<=Customers){
        // Wake the barber up if he is sleeping
        if(barber.s==0){
            barber.s=1;
        }

        // Cut the hair
        cut.s=1;
        cout<<"Customer "<<id<<" is getting a haircut\n\n";
        cut.s=0;

        id++;
    }

    // The barber goes to sleep after tending to all the customers
    barber.s=0;
}
```

```
int main(){
    cout<<"SOLUTION TO SLEEPING BARBER PROBLEM USING SEMAPHORE C++ IMPLEMENTATION\n";
    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";

    binarySemaphore barber;    // denotes if the barber is sleeping or awake
    binarySemaphore cut;       // semaphore to synchronise hair cutting
    countingSemaphore freeChairs;

    cout<<"Enter the number of free chairs: ";
    cin>>freeChairs.s;

    barber.s=0; // initially the barber is sleeping
    cut.s=0;

    // Suppose the customers visit the shop in groups. The group visits the barber only after all the
customers of first group have left the shop.
    while(1){
        int freeSpace=freeChairs.s;
        int customers=0;
        cout<<"Enter the number of customers entering the shop (enter 0 to exit): ";
        cin>>customers;

        if(customers==0){
            break;
        }

        // The first customer can always go to the sleeping barber
        int id=2;
        while(id<=customers){
            // Chair occupied by customer
            freeChairs.s--;
            id++;
            if(freeChairs.s==0){
                break;
            }

        }

        while(id<=customers){
            cout<<"Customer "<<id<<" returned back without getting a haircut\n";
            id++;
        }

        customer(barber, cut, (freeSpace-freeChairs.s+1));
        freeChairs.s=freeSpace;
    }

    return 0;
}
```

RESULT:

```
PS C:\Users\beadi\Desktop\OS LAB\Assignment 7> cd "c:\Users\beadi\Desktop\OS LAB\Assignment 7\" ;
rProblem }
SOLUTION TO SLEEPING BARBER PROBLEM USING SEMAPHORE C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009        Branch: IT


Enter the number of free chairs: 3
Enter the number of customers entering the shop (enter 0 to exit): 6
Customer 5 returned back without getting a haircut
Customer 6 returned back without getting a haircut
Customer 1 wakes the barber
Customer 1 is getting a haircut

Customer 2 is getting a haircut

Customer 3 is getting a haircut

Customer 4 is getting a haircut

The barber goes back to sleep

Enter the number of customers entering the shop (enter 0 to exit): 2
Customer 1 wakes the barber
Customer 1 is getting a haircut

Customer 2 is getting a haircut

The barber goes back to sleep

Enter the number of customers entering the shop (enter 0 to exit): 4
Customer 1 wakes the barber
Customer 1 is getting a haircut

Customer 2 is getting a haircut

Customer 3 is getting a haircut

Customer 4 is getting a haircut

The barber goes back to sleep

Enter the number of customers entering the shop (enter 0 to exit): 0
```