# ASSIGNMENT: CSE316

## INTEGRATED B.TECH.-M.TECH.

## in

## COMPUTER SCIENCE AND ENGINEERING

By


# Arun Singh

**11804263**

**Roll No: A07**

**Section: K18ZV**


Submitted to

**Priyanka Mittal**



**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

**Problem:**

**1) There are 3 student processes and 1 teacher process. Students are supposed to do their assignments and they need 3 things for that pen, paper and question paper. The teacher has an infinite supply of all the three things. One student has pen, another has paper, and another has question paper. The teacher places two things on a shared table and the student having the third complementary thing makes the assignment and tells the teacher on completion. The teacher then places another two things out of the three and again the student having the third thing makes the assignment and tells the teacher on completion. This cycle continues. WAP to synchronize the teacher and the students.**

## Explain the problem in terms of operating system concept?

**Explanation:**

For the solution of this problem I have taken 2d array of all the student processer and resources and initialized that array with 0. Then for the completion of this I have made 3 different students processed in 3 different function named stud1, stud2, stud3. which is being executed by single s_thread and one t_thread for execution of teacher process. User will get a menu to select any two out of three resources that are to be placed on shared table. If one process is completed there will be a message printed on the screen saying process is completed. When one process is executing no other student or teacher process will execute and for achieving this, I have used Mutex lock. When a process starts to execute it acquires the lock and when it completes the execution releases the lock. After completion of all the three processes the program will end.

## Write the algorithm for the proposed solution of the assigned problem.

**Explanation:**

1) Take all the student processes and resources in 2d array = 0.
2) Make 3 functions for student and one function for teacher process respectively.
3) Void stud1()

   {

   pthread_mutex_lock(&lck);

```
printf("\n \t choices Made = ''paper",' question_paper');
stud[1][4]=1;
printf("\n\t Student 1 has completed the assignment.");
pthread_mutex_unlock(&lck);
}
```

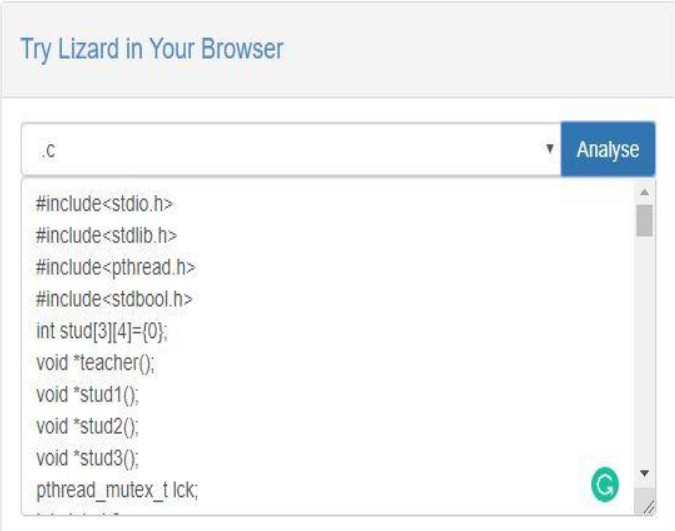
Same process for student2, student3.

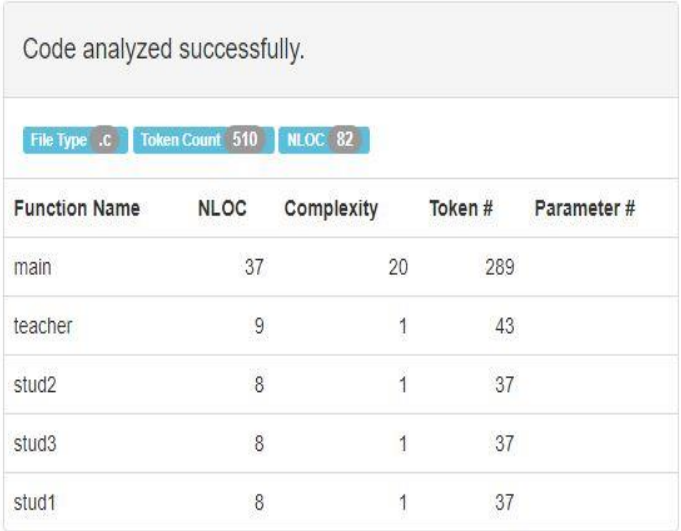
4) If one process completes there will be message of completion of that process.
5) No other process will execute while one process is being executing because of mutex lock.

## Calculate the complexity of the implemented algorithm.

The overall complexity of my code is order of $n^3$ .

And line wise complexity of the code is order of n for each if else statement and order of 1 for each function in the program.




| Function Name | NLOC | Complexity | Token # | Parameter # |
|---|---|---|---|---|
| main | 37 | 20 | 289 | |
| teacher | 9 | 1 | 43 | |
| stud2 | 8 | 1 | 37 | |
| stud3 | 8 | 1 | 37 | |
| stud1 | 8 | 1 | 37 | |

## Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

## Code Snippet:-

```c
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<stdbool.h>
int stud[3][4]={0};
void *teacher();
void *stud1();
void *stud2();
void *stud3();
pthread_mutex_t lck;
int ch1,ch2;
int r1,r2;
int main()
{
printf("\t\t\t---Welcome---\n");
    pthread_mutex_init(&lck,NULL);\
stud[1][1]=1;
    stud[2][2]=2;stud[3][3]=1;
    pthread_t t_thread;
    pthread_t s_thread;
printf("Resources Menu: \n\t\tPress '1' for pen\n\t\tPress '2' for paper \n\t\tPress '3' for
 question_paper \n");
    while(1)
{
if(stud[1][4]==1 && stud[2][4]==1 && stud[3][4]==1){break;}
pthread_create(&t_thread, NULL, teacher, NULL);
pthread_join(t_thread,NULL);
```

```c

if((ch1==1 && ch2==2 || ch2==1 && ch1==2 ) && stud[3][4]==0)
{
    pthread_create(&s_thread, NULL, stud3, NULL);
    pthread_join(s_thread,NULL);
}
else if((ch1==1 && ch2==3 || ch2==1 && ch1==3 ) && stud[2][4]==0)
{
pthread_create(&s_thread, NULL, stud2, NULL);
    pthread_join(s_thread,NULL);
}
else if((ch1==2 && ch2==3 || ch2==2 && ch1==3 ) && stud[1][4]==0)
{
    pthread_create(&s_thread, NULL, stud1, NULL);
pthread_join(s_thread,NULL);
}
else
{
    printf("\n\tError (007): try again.. with different choices.\n");
}
}
printf("\n\t----Done---\n");
}
```

```
51  void *teacher()
52  {
53  pthread_mutex_lock(&lck);
54      printf("\nFirst Resource on shared tabel:-\t");
55      scanf("%d",&ch1);
56      printf("Second Resource on shared tabel:-\t");
57      scanf("%d",&ch2);
58      pthread_mutex_unlock(&lck);
59  }
60  void *stud2()
61  {
62      pthread_mutex_lock(&lck);
63      printf("\nChoices Made = 'pen', 'question_paper'\n");
64      student[2][4]=1;
65      printf("\n\tStudent 2 has Completed the assignment. \n");
66      pthread_mutex_unlock(&lck);
67  }
68  void *stud3()
```

```
69  {
70      pthread_mutex_lock(&lck);
71      printf("\nChoices Made = 'pen', 'paper'\n");
72      student[3][4]=1;
73      printf("\n\tStudent 3 has Completed the assignment.\n");
74      pthread_mutex_unlock(&lck);
75  }
76  void *stud1()
77  {
78      pthread_mutex_lock(&lck);
79      printf("\nChoices Made = 'paper', 'question_paper'\n");
80      student[1][4]=1;
81      printf("\n\tStudent 1 has Completed the assignment.\n");
82      pthread_mutex_unlock(&lck);
83  }
```

## If you have implemented an additional algorithm to support the solution. Explain the need and use of the same.

**In the additional part I have solved the same problem with simple C compiler. In this there is no need of linex.**

**Code snippet –**

```c
#include<stdio.h>
#include<stdbool.h>
struct requirement
{
    bool pen ;
    bool paper ;
    bool question_paper ;
    bool all_three ;
};
int main()
{
    int n=3;
    struct requirement  s[n];
    s[0].pen=true;
    s[0].paper = false;
    s[0].question_paper = false;
    s[0].all_three= false;
    s[1].pen=false;
    s[1].paper = true;
    s[1].question_paper = false;
    s[1].all_three = false;
    s[2].pen=false;
    s[2].paper = false;
    s[2].question_paper = true;
    s[2].all_three = false  ;
    while(s[0].all_three==false||s[1].all_three==false||s[2].all_three==false)
    {
        int ch1,ch2;
        printf("\nResources:\n1.pen\n2.paper\n3.question paper\n Enter
        the two things which is to be placed on the shared table: ");
        scanf("%d%d",&ch1,&ch2);
        if(ch1==1 && ch2==2 && s[2].all_three==false)
        {
            s[2].all_three=true ;
            printf("Third Student has completed the task\n");
        }
        if(ch1==2 && ch2==3 && s[0].all_three==false)
        {
            s[0].all_three=true;
            printf("First Student has completed the task\n");
        }
        if(ch1==1 && ch2==3 && s[1].all_three==false)
        {
            s[1].all_three=true;
            printf("Second Student has completed the task\n");
        }
    }
    printf("All the students now have completed their respective tasks succesfully\n");
    return 0;
}
```

## Explain the boundary conditions of the implemented code.

BOUNDARY CONDITIONS ---

a) Boundary condition for the number of items one student can pick = 2.

b) Total number of process(students)=3.

c) Number of teacher process = 1.

d) Number of resources provided=3.

## Explain all the test cases applied on the solution of assigned problem.

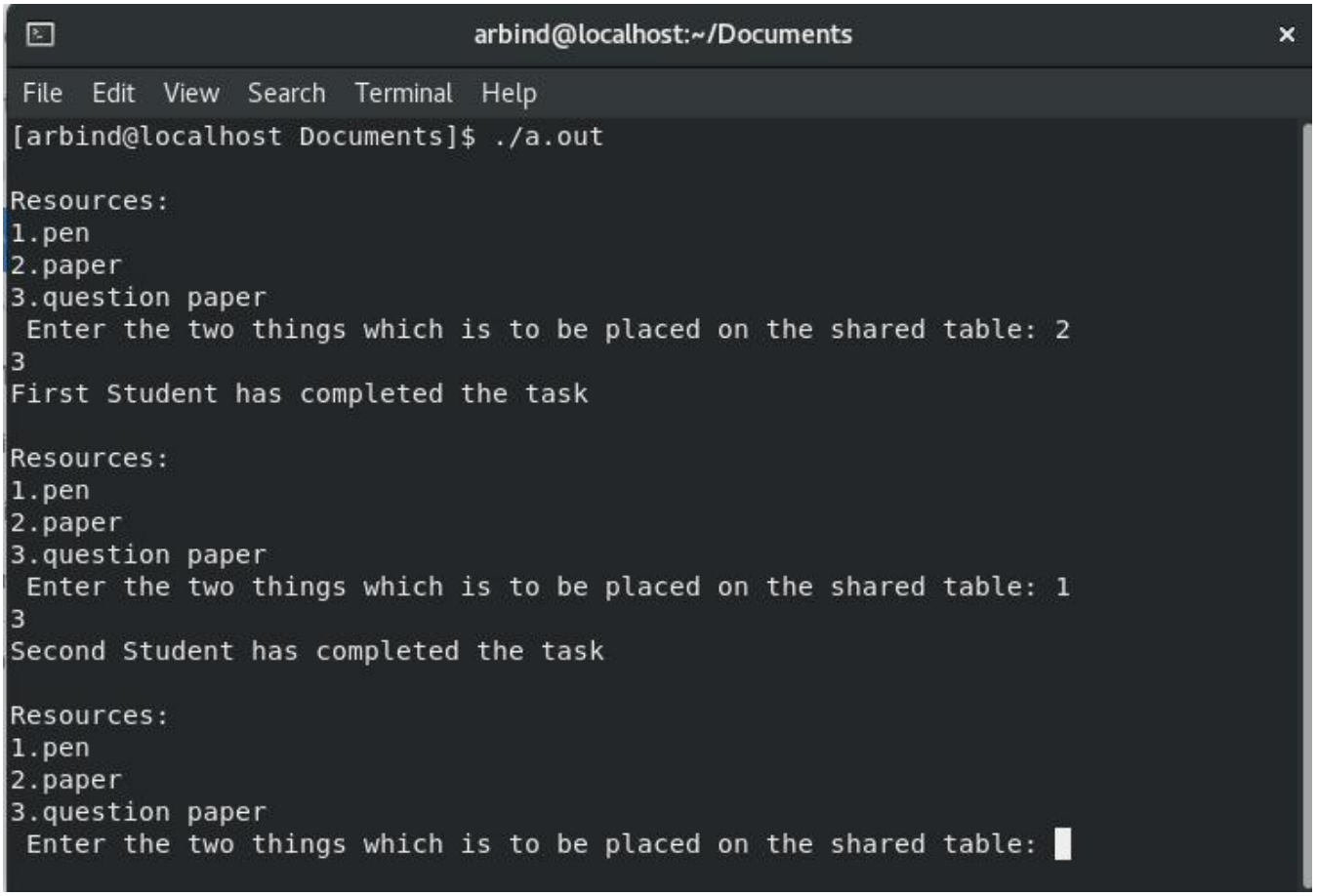
Resources provided

1. First student - pen
2. Second student - paper
3. Third student - question paper

If we provide the paper and question paper to the first student then first student will complete the task

Like shown in the output of the code

Test cases – 2,3 first will complete the task ; 1,3 2$^{nd}$ will complete the task.




## Have you made minimum of 5 revision of solution on GitHub ?

Yes I have made the 5 revision of the solution on GitHub.

GitHub link : https://github.com/savarun007/Oprating-system-Assignment-