ASSIGNMENT: CSE316

INTEGRATED B.TECH.-M.B.A.

in

COMPUTER SCIENCE AND ENGINEERING

By

Pratyaksh Rana

11801793

Roll No: B42

Section: K18ZV

Email: pratyakshrana2@gmail.com

GitHub Link: https://github.com/PratyakshRana/OSproject

Questions Assigned: 3 & 24

Submitted to

Mrs. Priyanka Mittal

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Description

I. In very first question we have to write a multithread program in c language which gives outputs as prime numbers. First the user will enter some input in the program after executing it, after giving the input, a separate thread will occur which prints all the prime numbers that are less than or equal to the input given by the user. For this question we have to have a brief knowledge about Threads concept in Operating system. Just like a process, a thread is also a sequence of instructions, but threads can be used to run many sub processes in parallel to carry out the task in smaller amount of time and unlike processes, threads share some part of their code with each other.

II. In second question, There are five processes P0, P1, P2, P3, P4 and 4 resources A, B, C, D. Available instances of resources, allocated instances of resources by the processes and maximum requirement of instances by the processes are also given. It is needed to write a program in C to check whether the system is in safe state or not from the deadlock. For this question one must have the knowledge of the topic known as DEADLOCKS. A deadlock occurs when multiple processes are holding resources required by other processes, neither process can complete its task and remains in deadlock.

Available				Processes	Allocation				Max			
A	B	C	D		A	B	C	D	A	B	C	D
1	5	2	0	P0	0	0	1	2	0	0	1	2
				P1	1	0	0	0	1	7	5	0
				P2	1	3	5	4	2	3	5	6
				P3	0	6	3	2	0	6	5	2
				P4	0	0	1	4	0	6	5	6

Functional Requirements

1:

- Program should use Threads
- Program should print prime numbers less than or equal to the user input
- Program should be in C programming language
- A separate thread must be created to carry out the remaining process
- Program should first ask for an input

2:

- Program should be in C programming language
- Program must check if the given system of processes is in safe state or not

These all constraints were satisfied by the program

Algorithms

A. Algo for finding prime numbers less than or equals to a given number:

Assume the number be N
For i = 1 till N:
    If i is having no divisors other than 1 and itself
    Print i

Complexity: $O(n^2)$

B. Algo for checking if the system is in safe state or not

X = Number of processes, A = Number of resources
Available[Aj] = Available instances of resource Aj
Maximum[Xi][Aj] = Maximum instances of resource Aj required by process Xi
Allocation[Xi][Aj] = Instances of Aj currently allocated to Xi
Need[Xi][Aj] = Remaining instances of Aj needed by Xj to complete its task
Finish[Xi] = Process Xi has finished its task or not

For passes 1 to X
(1): Initialize Finish[i] = false for i = 1 to n
(2): Find a process i such that
    (a): Finish[i] = false
    (b): Need of i <= Available
        If no such process then Go to step (4)
(3): Available = Available + Allocation of i
    Finish[i] = true
    Go to step (2)
(4): If Finish[i] = true for all i <= X
    Then the system is in Safe State

Complexity: $O(n^2m)$

Code

1:

```
#include<stdio.h>
#include<pthread.h>
void *prime(void *arg)
{
	int *n;
	n = (int *)arg;
	int count,i,j;
	for(i=2;i<=n[0];i++)
	{
		count = 0;
		for(j=1;j<=i;j++)
		{
			if(i%j==0)
			{
				count++;
			}
		}
		if(count==2)
		{
			printf("\n%d",i);
		}
	}
	return NULL;
}

int main()
{
	int n;
	printf("Enter a Number: ");
	scanf("%d",&n);
	printf("Prime Numbers Less than or Equals to %d are:",n);
```

```
        pthread_t thrd;
        pthread_create(&thrd,NULL,prime,(void *)&n);
pthread_join(thrd,NULL);
}
```

2:

```
#include <stdio.h> int
main()
{
        int i,j,n=5,m=4;
        int avail[4] = {1,5,2,0};
        int alloc[5][4] = {{0,0,1,2}, {1,0,0,0}, {1,3,5,4}, {0,6,3,2}, {0,0,1,4}};
        int max[5][4] = {{0,0,1,2},{1,7,5,0}, {2,3,5,6}, {0,6,5,2}, {0,6,5,6}};
int need[n][m];
        for ( i = 0 ; i < n ; i++ )
        {
                for ( j = 0 ; j < m ; j++ )
                {need[i][j] = (max[i][j]) - (alloc[i][j]);}
        }
        int isAvail;
int isSafe = 1;                 int
finish[5] = {0};
        int pass;
        for ( pass = 0 ; pass < n ; pass++)
        {
                for ( i = 0 ; i < n ; i++ )
                {
                        if ( finish[i] == 0 )
                        {
                                isAvail = 1;
                                for ( j = 0 ; j < m ; j++ )
                                {
                                        if ( need[i][j] > avail[j] )
                                        {isAvail = 0;}
                                }
                                if ( isAvail == 1 )
                                {
                                        for ( j = 0 ; j < m ; j++ )
                                        {avail[j] = avail[j] + alloc[i][j];}
                        finish[i] = 1;
```

```
                                }
                }
                }
        }

        for ( i = 0 ; i < n ; i++ )
        {
                if ( finish[i] == 0)
                {isSafe = 0;}
        }
        if ( isSafe == 1 )
        {printf("The System is in Safe State");}
        else
        {printf("The System is not in Safe State");} }
```

Test Cases

1:

For N = 7
Output: 2, 3,5,7

For N = 13
Output: 2, 3, 5, 7 ,11,13

For N = 19
Output: 2, 3, 5, 7, 11, 13,17,19

For N = 0
Output: No Output

For N = 31
Output: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ,31

2:
No test cases
Output will be Sytem is in safe state

GitHub

Link: https://github.com/PratyakshRana/OSproject