LOVELY PROFESSIONAL UNIVERSITY

Academic Task-3 (Operating System)

School of Computer Science and Engineering Faculty of Technology & Sciences

Name of the faculty member: Ashu

Course Code: CSE 316 Course Title: Operating System
Max. Marks: 30 Date of Allotment: 29/02/2020

Student Name: Manvir Kaur
Student ID: 11604523
Email Address: manvirk525@gmail.com
GitHub Link: https://github.com/manvir525/OperatingSystemProject
Code: https://github.com/manvir525/OperatingSystemProject/blob/master/SafeState.c

Problem:

Ques. 19. There are 5 processes and 3 resource types, resource A with 10 instances, B with 5 instances and C with 7 instances. Consider following and write a c code to find whether the system is in safe state or not?

Available			Processes	Allocation			Max		
A	B	C		A	B	C	A	B	C
3	3	2	P0	0	1	0	7	5	3
			P1	2	0	0	3	2	2
			P2	3	0	2	9	0	2
			P3	2	1	1	2	2	2
			P4	0	0	2	4	3	3

Description:

Considering a system with five processes P0 through P4 and three resources types A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. We must determine whether the new system state is safe. To do so, we need to execute Safety algorithm on the above given allocation chart.

Banker's Algorithm is a resource allocation and deadlock avoidance algorithm. This algorithm test for safety simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

In simple terms, it checks if allocation of any resource will lead to deadlock or not, OR is it safe to allocate a resource to a process and if not, then resource is not allocated to that process. Determining a safe sequence (even if there is only 1) will assure that system will not go into deadlock.

Banker's algorithm is generally used to find if a safe sequence exit or not. But here we will determine the total number of safe sequences and print all safe sequences.

Let **'n'** be the number of processes in the system and **'m'** be the number of resources types.

Available:

- It is a 1-d array of size **'m'** indicating the number of available resources of each type.
- Available [j] = k means there are **'k'** instances of resource type $\mathbf{R_j}$

Max:

- It is a 2-d array of size '**n*m'** that defines the maximum demand of each process in a system.
- Max [i, j] = k means process $\mathbf{P_i}$ may request at most **'k'** instances of resource type $\mathbf{R_j}$.

Allocation:

- It is a 2-d array of size **'n*m'** that defines the number of resources of each type currently allocated to each process.
- Allocation [i, j] = k means process $\mathbf{P_i}$ is currently allocated **'k'** instances of resource type $\mathbf{R_j}$

Need:

- It is a 2-d array of size **'n*m'** that indicates the remaining resource need of each process.
- Need [i, j] = k means process $\mathbf{P_i}$ currently allocated **'k'** instances of resource type $\mathbf{R_j}$
- Need [i, j] = Max [i, j] – Allocation [i, j]

$Allocation_i$ specifies the resources currently allocated to process P_i and $Need_i$ specifies the additional resources that process P_i may still request to complete its task.
Banker's algorithm consists of Safety algorithm and Resource request algorithm.

Safety Algorithm:

The algorithm for finding out whether or not a system is in a safe state can be described as follows:

1. Let Work and Finish be vectors of length 'm' and 'n' respectively.
   Initialize: Work= Available
   Finish [i]=false; for i=1, 2 ,…, n

2. Find an i such that both
   a) Finish [i]=false
   b) Need_i<=work
   if no such i exists goto step (4)

3. Work=Work + Allocation_i
   Finish[i]= true
   goto step (2)

4. If Finish[i]=true for all i,
   then the system is in safe state.

Safe sequence is the sequence in which the processes can be safely executed.

Time Complexity:

Overall Time Complexity = O(n*n*m)

where n = number of processes and m = number of resources.

Constraints given in the problem:

There are 5 processes and 3 resource types, resource A with 10 instances, B with 5 instances and C with 7 instances.

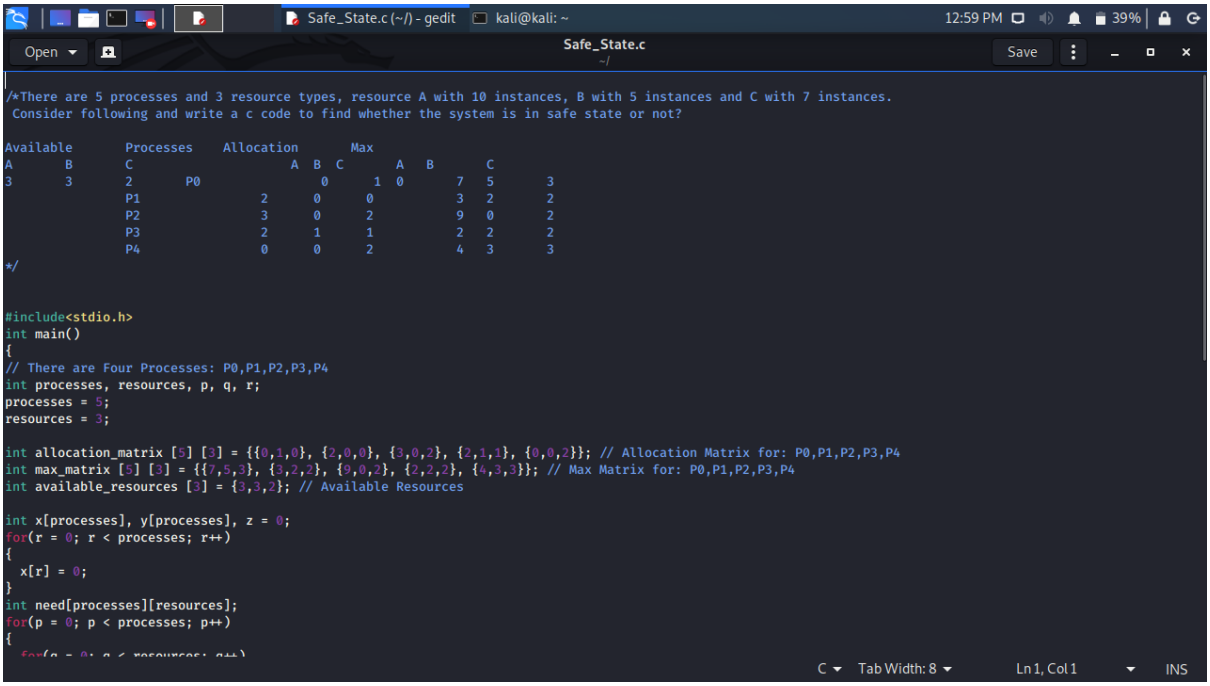
Boundary condition:

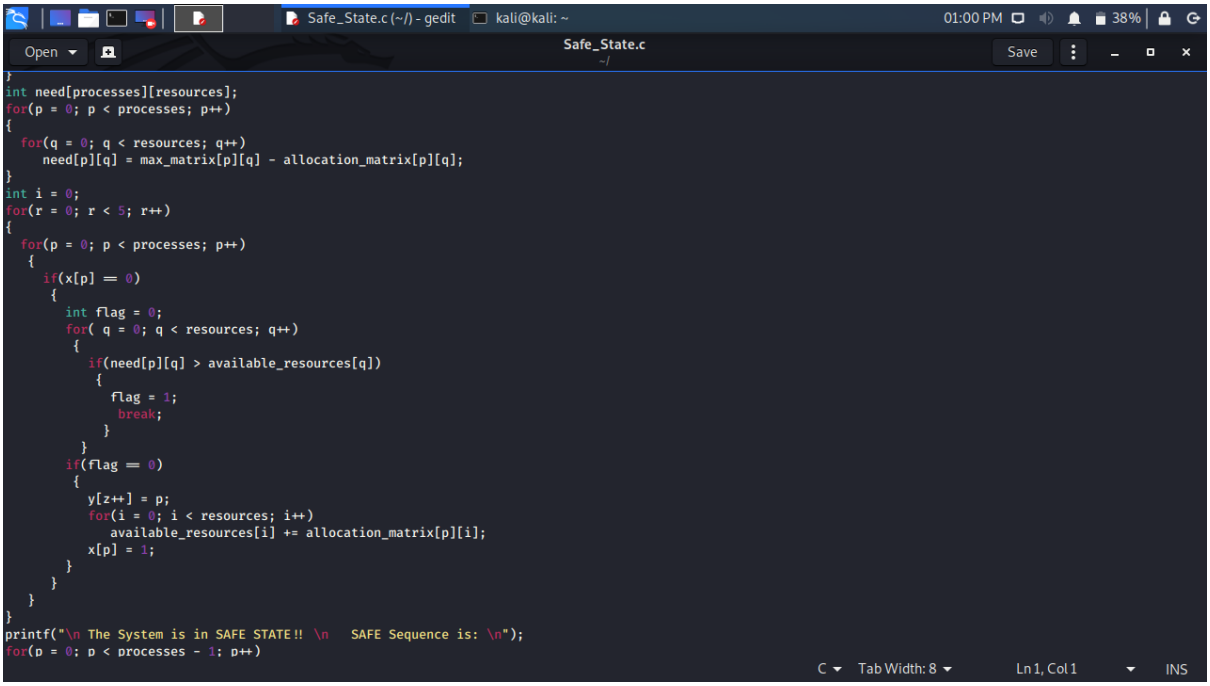
When a new process enters a system, it must declare the maximum number of instances of each resource type that may not exceed the total number of resources in the system.

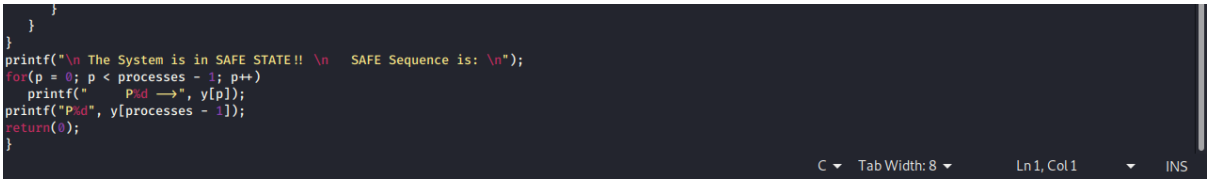
Test cases applied:

Available			Processes	Allocation			Max		
A	B	C		A	B	C	A	B	C
3	3	2	P0	0	1	0	7	5	3
			P1	2	0	0	3	2	2
			P2	3	0	2	9	0	2
			P3	2	1	1	2	2	2
			P4	0	0	2	4	3	3

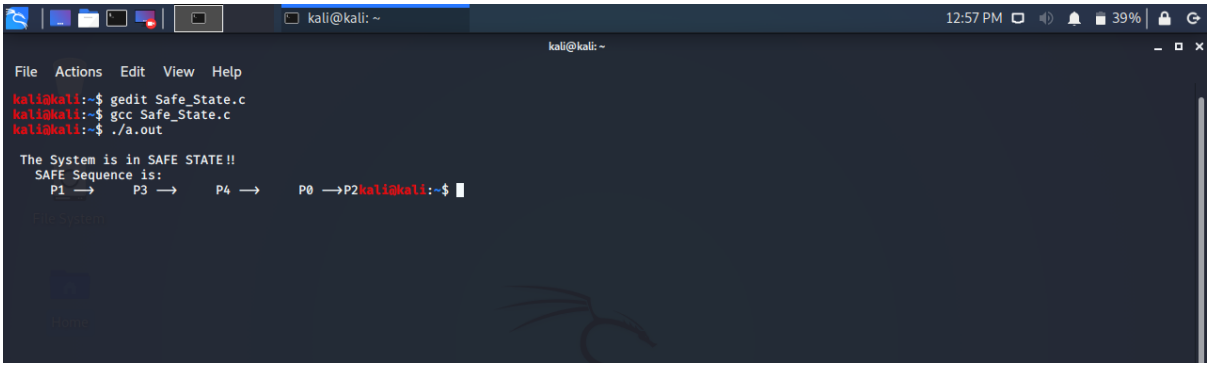
Code Snippet:







Output:



GitHub Link:

https://github.com/manvir525/OperatingSystemProject/blob/master/SafeState.c