Rajiv Gandhi Institute of Petroleum Technology, in Jais,
Amethi, Uttar Pradesh, India,

B.Tech Semester-vi: Operating system project

Project Title: **Deadlock Simulation**

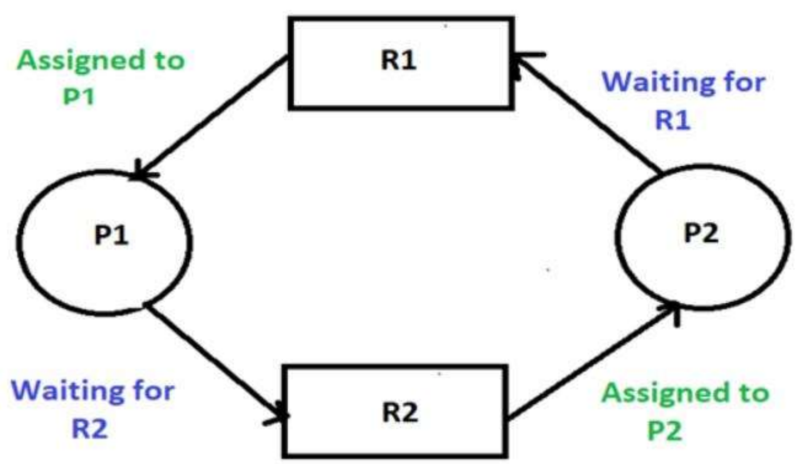
ROHIT KUMAR

20CS3053

- Problem definition:

Deadlock in Operating System is a situation that arises when a process in the computer waits for the resources to be released that are assigned to some other process.

Assume that Process 1 (P1) is holding Resource 1(R1) and waiting for Resource 2 (R2) which is acquired by process 2(P2), and process 2(P2) is waiting for resource 1(R1)



- Condition for deadlock occur

I. Mutual Exclusion: When two or more resources cannot be shared.

II. **Hold and wait:** A process is holding one or more resources and simultaneously waiting for other resources.

III. **No preemption:** A resource is not taken from the process until and unless the process releases the resources.

IV. **Circular wait:** Here a set of processes are waiting for each other in a circular form.

Banker’s Agorithm:

The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by 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

Implementation deadlock Simulation:

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, 3, 4….n
2) Find an i such that both
a) Finish[i] = false
b) Needi <= 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 a safe state

Resource-Request Algorithm

Let Requesti be the request array for process Pi. Requesti [j] = k means process Pi wants k instances of resource type Rj. When a request for resources is made by process Pi, the following actions are taken:

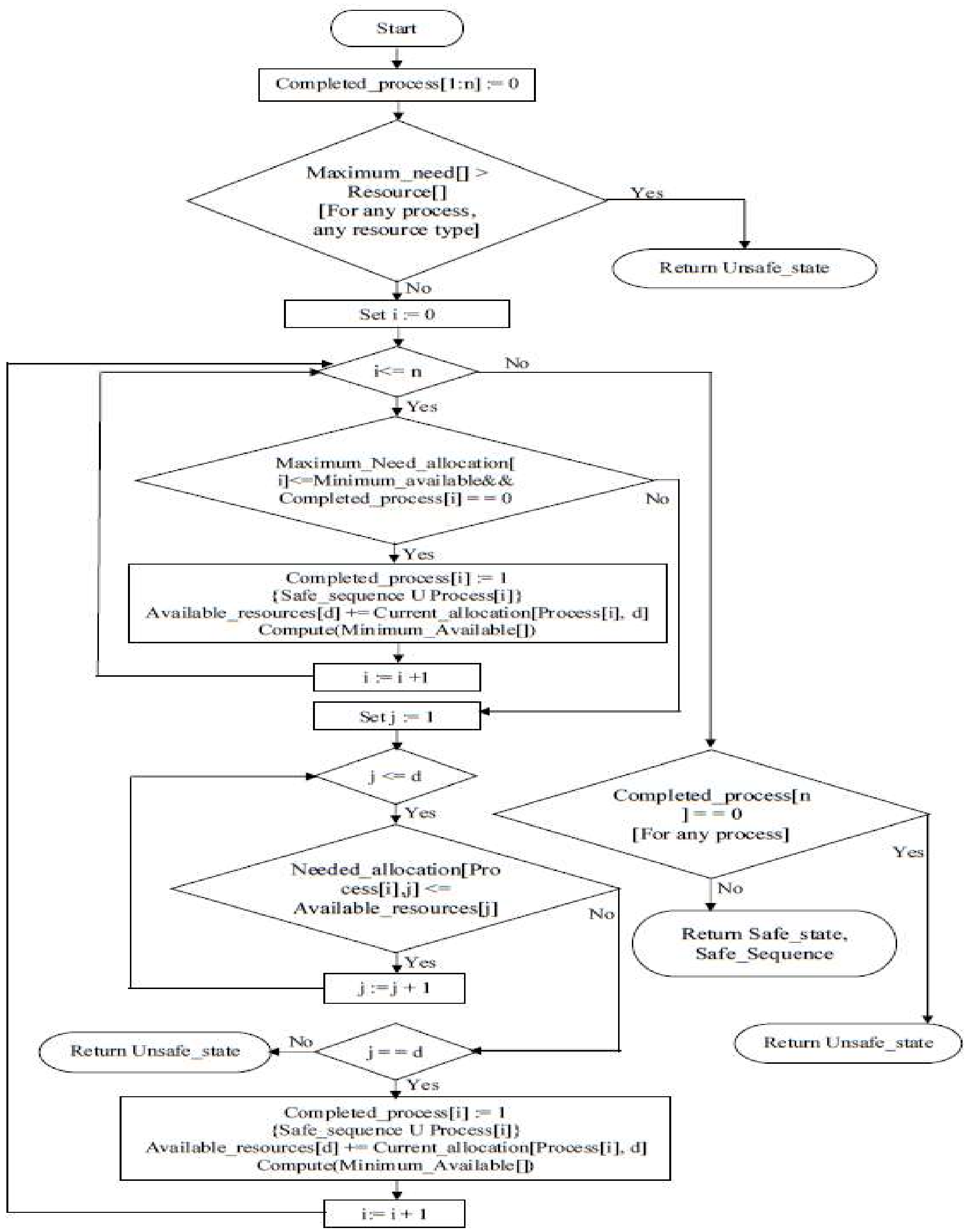
1) If Requesti <= Needi
Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2) If Requesti <= Available
Goto step (3); otherwise, Pi must wait, since the resources are not available.
3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as
follows:
Available = Available – Requesti
Allocationi = Allocationi + Requesti
Needi = Needi– Requesti

Example:

Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Flowchart


Start
Completed_process[1:n] := 0
Maximum_need[] > Resource[] [For any process, any resource type]
Yes
Return Unsafe_state
No
Set i := 0
i<= n
No
Yes
Maximum_Need_allocation[i]<=Minimum_available&& Completed_process[i] = = 0
No
Yes
Completed_process[i] := 1
{Safe_sequence U Process[i]}
Available_resources[d] += Current_allocation[Process[i], d]
Compute(Minimum_Available[])
i := i +1
Set j := 1
j <= d
Yes
Completed_process[n] = = 0 [For any process]
Yes
No
Return Safe_state, Safe_Sequence
Return Unsafe_state
Needed_allocation[Process[i],j] <= Available_resources[j]
No
Yes
j := j + 1
Return Unsafe_state
No
j = = d
Yes
Completed_process[i] := 1
{Safe_sequence U Process[i]}
Available_resources[d] += Current_allocation[Process[i], d]
Compute(Minimum_Available[])
i:= i + 1

★ Input:

```
C:\Users\kumar\Desktop\Deadlock_simulation\Banker's Algorithm Implementation.exe
How many sequence check for safe or not
1
sequence number=1
Number of processes
5
Number of resources
3
Allocation of resource Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Max resource need Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Needs of Resource for each process
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
How many Available Resource check for above Sequence
6
check Available Resources: 1
4 5 1
```

❖ ****RESULT*****

```
How many Available Resource check for above Sequence
6
check Available Resources: 1
4 5 1

<- - - - - - - - - - - - - - -RESULT- - - - - - - - - - - - -  - - - -  ->

The given sequence is
* * * * *SAFE Sequence* * * * *
 P3 -> P4 -> P1 -> P2 -> P0

check Available Resources: 2
5 1 0

<- - - - - - - - - - - - - - -RESULT- - - - - - - - - - - - -  - - - -  ->

The given sequence is
     * * * *not safe * * * *

check Available Resources: 3
3 3 2

<- - - - - - - - - - - - - - -RESULT- - - - - - - - - - - - -  - - - -  ->

The given sequence is
* * * * *SAFE Sequence* * * * *
 P1 -> P3 -> P4 -> P0 -> P2
```

```
check Available Resources: 4
6 6 6

<- - - - - - - - - - - - - - -RESULT- - - - - - - - - - - - -  - - - -  ->

The given sequence is
* * * * *SAFE Sequence* * * * *
 P1 -> P2 -> P3 -> P4 -> P0

check Available Resources: 5
9 0 0

<- - - - - - - - - - - - - - -RESULT- - - - - - - - - - - - -  - - - -  ->

The given sequence is
     * * * *not safe * * * *

check Available Resources: 6
7 6 5

<- - - - - - - - - - - - - - -RESULT- - - - - - - - - - - - -  - - - -  ->

The given sequence is
* * * * *SAFE Sequence* * * * *
 P0 -> P1 -> P2 -> P3 -> P4


--------------------------------
Process exited after 232.4 seconds with return value 0
Press any key to continue . . .
```

➢ **Conclusion**

Deadlock simulation is easly show sequence of process is safe state or not.As the processes enter the system, they must predict the maximum number of resources needed which is not impractical to determine. the number of processes remain fixed which is not possible in interactive systems. This algorithm requires that there should be a fixed number of resources to allocate. If a device breaks and becomes suddenly unavailable the algorithm would not work.Overhead cost incurred by the algorithm can be high when there are many processes and resources because it has to be invoked for every processes.

◊ Reference

Dr.Niraj kumar faculty of Operating system in RGIPT jais,amethi. https://www.linkedin.com/in/niraj-kumar-189276a2/?originalSubdomain=in

Operating System :Internal and System Design, 9th edition, William Stallings, Pearson Publication.

https://www.geeksforgeeks.org/bankers-algorithm-in-operating-system-2