SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Academic year	2023-24	Program	BTECH CS
Semester	V	Year	III
Course Title	Operating Systems	Name of the Faculty	Prof. Ishani Saha

PART A
(PART A : TO BE REFFERED BY STUDENTS)

Experiment No. 07

A.1—Aim:
Study of Deadlock and Implementation of Banker's Algorithm

A.2--- Prerequisite:
Concepts & Principles of Deadlock.

A.3--- Outcome:
After successful completion of this experiment students will be able to:
1. Understand the concept of Deadlock, Deadlock avoidance & Deadlock Prevention.
2. Implement Banker's Deadlock Avoidance Algorithm

A.4--- Theory:

Deadlock: A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

The earliest computer operating systems ran only one program at a time. All of the resources of the system were available to this one program. Later, operating systems ran multiple programs at once, interleaving them. Programs were required to specify in advance what resources they needed so that they could avoid conflicts with other programs running at the same time. Eventually some operating systems offered dynamic allocation of resources. Programs could request further allocations of resources after they had begun running. This led to the problem of the deadlock.

Bankers Algorithm

The Banker's algorithm, sometimes referred to as the avoidance algorithm, is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

The algorithm was developed in the design process for the THE operating system. When a new process enters a system, it must declare the maximum number of instances of each resource type that it may ever claim; clearly, that number may not exceed the total number of resources in the

Academic year	2023-24	Program	BTECH CS
Semester	V	Year	III
Course Title	Operating Systems	Name of the Faculty	Prof. Ishani Saha

system. Also, when a process gets all its requested resources it must return them in a finite amount of time.

A.5--- Procedure:
Task:
1. Study Deadlock.
2. Implement Banker's Algorithm for deadlock avoidance.
3. Save and close the file and name it as **EXP7_ your Roll no.**

-

SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Academic year	2023-24	Program	BTECH CS
Semester	V	Year	III
Course Title	Operating Systems	Name of the Faculty	Prof. Ishani Saha

PART B
(PART B: TO BE COMPLETED BY STUDENTS)

Roll No: B091	Name: Yash Korla
Class: BTech Computer Engineering	Batch: B2
Date of Experiment: 23-09-2023	Date of Submission: 24-09-2023
Grade:	

B.1 Work done by student

- **Code**

```
n = 5
m = 3

alloc = [[0, 1, 0], [2, 0, 0],
        [3, 0, 2], [2, 1, 1], [0, 0, 2]]

max = [[7, 5, 3], [3, 2, 2],
      [9, 0, 2], [2, 2, 2], [4, 3, 3]]

avail = [3, 3, 2]

f = [0]*n
ans = [0]*n
ind = 0
for k in range(n):
   f[k] = 0

need = [[0 for i in range(m)]for i in range(n)]
for i in range(n):
   for j in range(m):
      need[i][j] = max[i][j] - alloc[i][j]
y = 0
for k in range(5):
   for i in range(n):
      if (f[i] == 0):
         flag = 0
         for j in range(m):
            if (need[i][j] > avail[j]):
               flag = 1
               break

         if (flag == 0):
```

Academic year	2023-24	Program	BTECH CS
Semester	V	Year	III
Course Title	Operating Systems	Name of the Faculty	Prof. Ishani Saha

```
            ans[ind] = i
            ind += 1
            for y in range(m):
                avail[y] += alloc[i][y]
            f[i] = 1

print("Following is the SAFE Sequence")

for i in range(n - 1):
    print(" P", ans[i], " ->", sep="", end="")
print(" P", ans[n - 1], sep="")
```

- **Output**

Following is the SAFE Sequence
P1 -> P3 -> P4 -> P0 -> P2

B.2 Conclusion:
Studied Deadlock and Implemented Banker's Algorithm.

B.3 Questions of Curiosity:
Q1. Determine if the system is in safe state for the implemented example?
Ans: Yes, the system is in safe state as all the are finished.

Q2. Also write down the safe sequence for the implemented example.
Ans: P1 -> P3 -> P4 -> P0 -> P2