Banker's Algorithm

1 Problem Statement

Write a Java program to implement Banker's Algorithm

2 Learning Objectives

In this assignment, students will :

1. Understand Banker's algorithm

2. Understand Deadlocks

3 Learning Outcomes

After completion of this assignment, students will be able to :

1. Implement Banker's algorithm using OOP concepts

4 Requirements

Hardware : 64-bit 2.8 GHz processor, 4 GB RAM

Software : 64-bit OS, JDK

5 Theory

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. There are three ways to handle deadlock 1) Deadlock prevention or avoidance: The idea is to not let the system into deadlock state. One can zoom into each category individually, Prevention is done by negating one of above mentioned necessary conditions for deadlock. Avoidance is kind of futuristic in nature. By using strategy of "Avoidance", we have to make an assumption. We need to ensure that all information about resources which process WILL need are known to us prior to execution of the process. We use Banker's algorithm (Which is in-turn a gift from Dijkstra) in order to avoid deadlock.

2) Deadlock detection and recovery: Let deadlock occur, then do preemption to handle it once occurred.

3) Ignore the problem all together: If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

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. Banker's algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it.

In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would try to be in safe state always.

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

6 Output

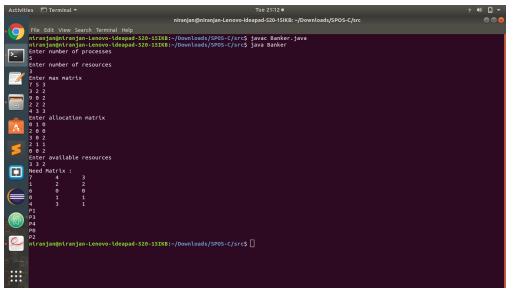

Figure 1: Output

7 Conclusion

Hence, we learnt how to use Banker's algorithm to determine safe execution sequence of the processes.