Exp No:7 Title of the Exercise: Bankers Algorithm

Date:26/09/2022

Aim: --

To implement Banker Algorithm

Procedure: --

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 come 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.

CODE: --

# P0, P1, P2, P3, P4 are the Process names here

n = 5 # Number of processes

m = 3 # Number of resources

# Allocation Matrix

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

# MAX Matrix

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

avail = [3, 3, 2] # Available Resources

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):

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: --



the system will skip process 0 and move to process 1. Because each process holds some resource and if the resource is not enough, the process can’t be completed. Since the system doesn’t have enough resources for process 0, it cannot complete that process. Hence the deadlock condition occurs. Once a process is completed, the system will regain the allocated resources. AS a result, there will be enough resources to allocate for upcoming processes and deadlock conditions can be avoided. Therefore, the system allocates resources in the order P1, P3, P4, P0, P2.

RESULT: --

We have successfully implement Banker’s algorithm.