**Branch :- Computer Sci. & Engg. Class :- Final Year**

**Subject :-System and Software Security Lab manual Sem :- VII**

**Teacher Manual**

**PRACTICAL NO 06**

**Practical no. 2**

**AIM**:- Interpret Bankers algorithm for Deadlock Avoidance.

**S/W REQUIRED:-** GCC Compiler

**THEORY:-**

The Banker’s algorithm is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra.

**Banker’s Algorithm working principle:**

It tests for safety by simulating the allocation of predetermined maximum possible amounts of all

resources, and then makes a “s-state” check to test for possible deadlock conditions for all other

pending activities, before deciding whether allocation should be allowed to continue.

**Characteristics of Banker's Algorithm**

1. If any process requests for a resource, then it has to wait.
2. This algorithm consists of advanced features for maximum resource allocation.
3. There are limited resources in the system we have.
4. In this algorithm, if any process gets all the needed resources, then it is that it should return the resources in a restricted period.

**Algorithm**

A safety algorithm is an algorithm used to find whether or not a system is in its safe state. The

algorithm is as follows:

1. Let Work and Finish be vectors of length m and n, respectively. Initially,

Work = Available

Finish[i] =false for i = 0, 1, ... , n - 1.

This means, initially, no process has finished and the number of available resources is represented by

the Available array.

1. Find an index i such that both

a)Finish[i] ==false

b)Needi <= Work

If there is no such i present, then proceed to step 4.

1. Perform the following:

Work = Work + Allocationi

Finish[i] = true

Go to step 2.

When an unfinished process is found, then the resources are allocated and the process is marked

finished. And then, the loop is repeated to check the same for all other processes.

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

That means if all processes are finished, then the system is in safe state.

This algorithm may require an order of mxn² operations in order to determine whether a state is safe or not.

**Program:**

#include <stdio.h> int main() {

// P0, P1, P2, P3, P4 are the names of Process

int n, r, i, j, k;

n = 5; // Indicates the Number of processes

r = 3; //Indicates the Number of resources

int alloc[5][3] = { { 0, 0, 1 }, // P0 // This is Allocation Matrix

{ 3, 0, 0 }, // P1

{ 1, 0, 1 }, // P2

{ 2, 3, 2 }, // P3

{ 0, 0, 3 } }; // P4

int max[5][3] = { { 7, 6, 3 }, // P0 // MAX Matrix

{ 3, 2, 2 }, // P1

{ 8, 0, 2 }, // P2

{ 2, 1, 2 }, // P3

{ 5, 2, 3 } }; // P4

int avail[3] = { 2, 3, 2 }; // These are Available Resources

int f[n], ans[n], ind = 0;

for (k = 0; k < n; k++) {

f[k] = 0;

}

int need[n][r];

for (i = 0; i < n; i++) {

for (j = 0; j < r; j++)

need[i][j] = max[i][j] - alloc[i][j];

}

int y = 0;

for (k = 0; k < 5; k++) {

for (i = 0; i < n; i++) {

if (f[i] == 0) {

int flag = 0;

for (j = 0; j < r; j++) {

if (need[i][j] > avail[j]){

flag = 1;

break;

}

}

if (flag == 0) {

ans[ind++] = i;

for (y = 0; y < r; y++)

avail[y] += alloc[i][y];

f[i] = 1;

}

}

}

}

printf("Th SAFE Sequence is as follows\n");

for (i = 0; i < n - 1; i++)

printf(" P%d ->", ans[i]);

printf(" P%d", ans[n - 1]);

return (0);

}

### Output:

Here is the output of the above program:

IMG_256

**CONCLUSION:** Thus, we have interpreted Bankers algorithm for Deadlock Avoidance