|  |
| --- |
| **Subject :Operating System Sub Teacher:Prof.S.S.Shethe**  **Class: S.E. Computer Engg Roll no :-**  **Practical No.: Date:** |

**Title:** Write A Program For Banker’s Algorithm .

**THEORY:**

This is a C program for Banker’s algorithm for finding out the safe sequence. Banker’s algorithm is used to schedule processes according the resources they need. It is very helpful in deadlock Handing. Banker’s algorithm produces a safe sequence as a output if all the resources can be executed and return error if no safe sequence of the processes available.

Example on Bankers Algorithm

Consider a system consist of five process (p1,p2,p3,p4,p5) and three resources (p1,p2,p3,p4,p5) type R1 has to instances resources type R2 has 5 instances and R3 has 7 instances

Process Allocation max available

R1,R2,R3,R1,R2,R3,R1,R2,R3

P1 0 1 0 1 5 3 3 2

P2 2 0 0 3 2

P3 3 0 2 9 0

P4 2 1 1 2 2

P5 0 0 2 4 3

Constant of the matrix need is calculated as need= max-allocation

Process Need

R1 R2 R3

P1 7 4 3

P2 1 2 2

P3 6 0 0

P4 0 1 1

P5 4 3 1

Safe sequences:- Safe sequences is calculated as follows

1) Need of each process is compared with available if need i< available than the resources an allowed to that process & process will relax the resources.

2) IF need is greater than available next process need is take for comparison

3) In the above example needs as process p1 (7,4,3) and available is (3,3,2) so <= available

4) Need for process p2 is (1,2,2) and available

(3,32) so need <=available

(1,2,,2) <= (3,3,2)=True

Then finish [i] = true

Request of P2 is granted a process   
P2 is release the resources so the system

work=work+allocation

=(3,3,2)+(2,0,,0)

=(5,3,2)

finish [i]=true

5) next process P3 need (6,0,0) is compare the need > available = false

(6,0,0) > (5,3,2)

6) Process P1 need (0,1,1) is compared with available (5,3,2)

need < available

(0,1,1) < (5,3,2)= True

available =available+ allocation

=(5,3,2) + (2,1,1)

=(7,4,3)

Than process P5 need (4,3,1) is compared with available (7,4,3)

need < available

(4,3,1) < (7,4,3)= true

available=available + allocation

(7,,4,3) + (0,0,2)

=(7,4,3)

7) Process P3 need (7,4,3) and available (7,8,5) If this request to granted then the system may be in the deadlock state after grantly the request is granted then the system is (0,0,2) so the system is in an safe state

8) Process P3 need is (6,0,0) and available (7,4,5) need < available-request

(6,0,0) < (7,4,3) = true

=(7,4,5) + (3,0,2)

=(10,4,7)Experiment No- 5

**Conclusion: -**

In this way we learnt how to implement the Bankers Algorithm.

/\*\*\*\*\*\*\*\*\*\*\* PROGRAM FOR BANKER'S ALGORITHM \*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

//#include<conio.h>

void main()

{

intalloc[3][3],max[3][3],inst[3],need[3][3],avail[3];

inti,j,work[3],proc[3],temp,finished=3,loop=10,last;

//clrscr();

printf("\n enter the allocation table for 3 process & 3 resource:\n");

for(i=0;i<3;i++)

{

printf("process%d:",i+1); //for input resource allocation

for(j=0;j<3;j++)

scanf("%d",&alloc[i][j]);

}

printf("\n enter the maximum resources table for 3 process:\n");

for(i=0;i<3;i++)

{

printf("process%d:",i+1);//for input max resource allocation

for(j=0;j<3;j++)

scanf("%d",&max[i][j]);

}

printf("\n enter total no of instances available for each resources:\n");

for(i=0;i<3;i++)

{

printf("process%c:",65+i); //for availability of resources

scanf("%d",&inst[i]);

}

printf("\n resources needed for 3 processes:");

printf("\n\t\tA\tB\tC\n-----------------------");

for(i=0;i<3;i++)

{

printf("\n process%d:\t",i+1);

for(j=0;j<3;j++)

{

need[i][j]=max[i][j]-alloc[i][j];//find need matrix

printf("%d\t",need[i][j]);

}

}

printf("\npresently available instance for each resource:\nA\tB\tC\n");

for(i=0;i<3;i++)

{

printf("\n process%d:\t",i+1);

temp=0;

for(j=0;j<3;j++)

temp+=alloc[j][i]; //check current status of resource

j=0;

last=finished;

while(finished)

{

if(need[j][0]<avail[0]&&need[j][1]<avail[1]&&need[j][2]<avail[2])

{

printf("\n"); //check deadlock occurance

for(i=0;i<3;i++)

{

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

}

printf("the sequence is p%d",j);

finished--;

temp=0;

}

j++;

if(j>=3)

j=0;

if(!temp)

{

last=finished+1;

temp=1;

}

if(finished==last&&loop!=0)

loop--;

if(!loop)

break;

}

if(finished)

printf("\n system is not running under safe condition");

else

printf("\n system is running under safe condition");

//getch();

}

}

**OUTPUT=>**

Enter the allocation table for 3 process& 3 resource:

Process1:2 1 0

Process2:1 1 0

Process3:2 1 2

Enter the maximum required resource table for 3 process:

Process1:5 3 2

Process2:3 2 2

Process3:3 1 5

Enter total number of instances available for each resource:

For Resource A:

For Resource B:For Resource C:

Resource Needed For 3 Process:

A B C

..................

Process 1: 3 2 2

Process 2: 2 1 2

Process 3: 1 0 3

Presently Available Instances For Each Resource:

A B C

The sequence is p0

The sequence is p1

The sequence is p2

System is running under safe condotions