**BANKERS ALGORITHM FOR DEADLOCK AVOIDANCE**

SANJANA S NAIR

CSE C

33

#include<stdio.h>

void main()

{

int available[10], allocated[30][30], need[30][30], max[30][30], m, n, finish[20], req[20], w[10], i, j, temp[10], p, k, f=0, count=0;

printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter the number of resource types: ");

scanf("%d", &m);

printf("Enter the allocation matrix:\n");

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

{

finish[i] = 0;

printf("Process %d:\n", i);

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

{

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

}

}

printf("Enter the Maximum Resource matrix:\n");

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

{

printf("Process %d:\n", i);

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

{

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

}

}

printf("Enter the available resources:\n");

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

{

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

}

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

{

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

{

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

}

}

printf("\n-------------------Resource Allocation Table----------------------\n");

printf("Process\tAllocated\tMaximum\t\tNeed\t\tAvailable\n");

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

{

printf("P%d\t", i);

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

{

printf("%d ", allocated[i][j]);

}

printf("\t\t");

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

{

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

}

printf("\t\t");

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

{

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

}

printf("\t\t");

if(i == 0)

{

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

{

printf("%d ", available[j]);

}

}

printf("\n");

}

printf("\nEnter the requesting process number : ");

scanf("%d",&p);

printf("\nEnter the requesting instances : \n");

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

{

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

}

f=0;

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

{

if(req[i]>available[i] && req[i]>need[p][i])

{

f++;

break;

}

}

if(f==0)

{

printf("Request Status : Successfull\n");

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

{

available[i]-=req[i];

need[p][i]-=req[i];

allocated[p][i]+=req[i];

}

printf("\n----------Modified Resource allocation table----------\n");

printf("Pro\tAllocated\tMaximum\tNeed\tAvailable\n");

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

{

printf("P%d\t", i);

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

{

printf("%d ", allocated[i][j]);

}

printf("\t\t");

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

{

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

}

printf("\t\t");

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

{

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

}

printf("\t\t");

if(i == 0)

{

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

{

printf("%d ", available[j]);

}

}

printf("\n");

}

int c=0;

f=0;

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

{

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

{

if(finish[j]==0)

{

for(k=0; k<m; k++)

{

if(available[k]<need[j][k])

{

f=1;

break;

}

}

if(f==0)

{

printf("\n[Process %d] : Executed\n",j);

w[c]=j;

c++;

finish[j]=1;

count++;

printf("Available : ");

for(k=0; k<m; k++)

{

available[k]=available[k]+allocated[j][k];

printf("%d ",available[k]);

}

}

}

f=0;

}

}

if(count==n)

{

printf("\n\n------DeadLock Free-------");

printf("\nSafety Sequence : ");

printf("p%d",w[0]);

for(i=1; i<c; i++)

{

printf(" --> p%d ",w[i]);

}

}

else

{

printf("\n------DeadLock Condition-------\n");

}

printf("\n");

}

else

printf("\nRequest status : Fail\n");

}

**SAMPLE OUTPUT**

Enter number of processes: 5

Enter number of resources: 3

Process 0 details.

Enter allocation: 0 1 0

Enter max requirements: 7 5 3

Process 1 details.

Enter allocation: 2 0 0

Enter max requirements: 3 2 2

Process 2 details.

Enter allocation: 3 0 2

Enter max requirements: 9 0 2

Process 3 details.

Enter allocation: 2 1 1

Enter max requirements: 2 2 2

Process 4 details.

Enter allocation: 0 0 2

Enter max requirements: 4 3 3

Enter available: 3 3 2

New resource request:

Enter pid: 1

Enter resource request: 1 0 2

P1 is visited Available mem: ( 5 3 2 )

P3 is visited Available mem: ( 7 4 3 )

P4 is visited Available mem: ( 7 4 5 )

P0 is visited Available mem: ( 7 5 5 )

P2 is visited Available mem: ( 10 5 7 )

System is in Safe state

Safe Sequence: P1 P3 P4 P0 P2

Name Max Allocation Need

P0 7 5 3 0 1 0 7 4 3

P1 3 2 2 3 0 2 0 2 0

P2 9 0 2 3 0 2 6 0 0

P3 2 2 2 2 1 1 0 1 1

P4 4 3 3 0 0 2 4 3 1