|  |  |
| --- | --- |
| **operating**  **SYSTEMS**  **SWE2007** | **LAB ASSESMENT** TERALA SRUJAN-18MIS7202 |

1.Implementation of Deadlock Prevention Using Banker’s Algorithm.

#include<stdio.h>

int max[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n,r;

void input();

void show();

void cal();

int main()

{

int i,j;

printf(" --------TERALA SRUJAN 18MIS7202------------\n");

printf(" ------------Banker's Algorithm------------\n");

input();

show();

cal();

getch();

return 0;

}

void input()

{

int i,j;

printf("Enter the no of Processes\t");

scanf("%d",&n);

printf("Enter the no of resources instances\t");

scanf("%d",&r);

printf("Enter the Max Matrix\n");

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

{

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

{

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

}

}

printf("Enter the Allocation Matrix\n");

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

{

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

{

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

}

}

printf("Enter the available Resources\n");

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

{

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

}

}

void show()

{

int i,j;

printf("Process\t Allocation\t Max\t Available\t");

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

{

printf("\nP%d\t ",i+1);

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

{

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

}

printf("\t");

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

{

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

}

printf("\t");

if(i==0)

{

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

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

}

}

}

void cal()

{

int finish[100],temp,need[100][100],flag=1,k,c1=0;

int safe[100];

int i,j;

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

{

finish[i]=0;

}

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

{

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

{

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

}

}

printf("\n");

while(flag)

{

flag=0;

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

{

int c=0;

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

{

if((finish[i]==0)&&(need[i][j]<=avail[j]))

{

c++;

if(c==r)

{

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

{

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

finish[i]=1;

flag=1;

}

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

if(finish[i]==1)

{

i=n;

}

}

}

}

}

}

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

{

if(finish[i]==1)

{

c1++;

}

else

{

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

}

}

if(c1==n)

{

printf("\n The system is in safe state");

}

else

{

printf("\n Process are in dead lock");

printf("\n System is in unsafe state");

}

}

Output:

---------TERALA SRUJAN 18MIS7202-----------

------------Banker's Algorithm------------

Enter the no of Processes 5

Enter the no of resources instances 3

Enter the Max Matrix

4

5

6

4

5

9

1

2

3

4

3

2

1

4

5

Enter the Allocation Matrix

4

2

3

5

1

2

3

1

5

6

3

2

4

6

7

Enter the available Resources

7

0

2

Process Allocation Max Available

P1 4 2 3 4 5 6 7 0 2

P2 5 1 2 4 5 9

P3 3 1 5 1 2 3

P4 6 3 2 4 3 2

P5 4 6 7 1 4 5

P3->P2->P0->P1->P4->

The system is in safe state

2.   Implementation of Page Replacement (Least Recent Use) Algorithm.

#include<stdio.h>

main ()

{

int temp[20], page[50], c = 0, c1, d, f, i, j, k = 0, n, r, t, b[20], c2[20];

printf ("-------TERALA SRUJAN 18MIS7202 ------");

printf ("\n");

printf ("-------LEASE PAGE REPLACEMENT ALGORITHM ------");

printf ("\n");

printf ("Enter no of pages:");

scanf ("%d", &n);

printf ("Enter the page values:");

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

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

printf ("Enter no of frames:");

scanf ("%d", &f);

temp[k] = page[k];

printf ("\n\t%d\n", temp[k]);

c++;

k++;

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

{

c1 = 0;

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

{

if (page[i] != temp[j])

c1++;

}

if (c1 == f)

{

c++;

if (k < f) {

temp[k] = page[i];

k++;

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

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

printf ("\n");

}

else

{

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

{

c2[r] = 0;

for (j = i - 1; j < n; j--)

{

if (temp[r] != page[j])

c2[r]++;

else

break;

}

}

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

b[r] = c2[r];

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

{

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

{

if (b[r] < b[j])

{

t = b[r];

b[r] = b[j];

b[j] = t;

}

}

}

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

{

if (c2[r] == b[0])

temp[r] = page[i];

printf ("\t%d", temp[r]);

}

printf ("\n");

}

}

}

printf ("\nThe no of page faults is %d", c);

}

Output:

1

-------TERALA SRUJAN 18MIS7202 ------

-------LEASE PAGE REPLACEMENT ALGORITHM ------

Enter no of pages:20

Enter the page values:

7

0

1

2

0

3

0

4

2

3

0

3

2

1

2

0

1

7

0

1

Enter no of frames:4

7

7 0

7 0 1

7 0 1 2

3 0 1 2

3 0 4 2

3 0 1 2

7 0 1 2

The no of page faults is 8

3.  Implementation of FCFS Disk Scheduling algorithm.

#include<stdio.h>

void

main ()

{

int queue[100], n, head, i, j, k, seek = 0, diff;

float avg;

printf ("----- TERALA SRUJAN 18MIS7202 ------\n");

printf ("----- FCFS Disk Scheduling Algorithm ------\n");

printf ("Enter the size of Queue\t");

scanf ("%d", &n);

printf ("Enter the Queue\t");

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

{

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

}

printf ("Enter the initial head position\t");

scanf ("%d", &head);

queue[0] = head;

printf ("\n");

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

{

diff = abs (queue[j + 1] - queue[j]);

seek += diff;

printf ("Move from %d to %d with Seek %d\n", queue[j], queue[j + 1], diff);

}

printf ("\nTotal Seek Time is %d\t", seek);

avg = seek / (float) n;

printf ("\nAverage Seek Time is %f\t", avg);

}

Output:

----- TERALA SRUJAN 18MIS7202 ------

----- FCFS Disk Scheduling Algorithm ------

Enter the size of Queue 10

Enter the Queue 15

180

52

43

14

68

32

25

30

54

Enter the initial head position 50

Move from 50 to 15 with Seek 35

Move from 15 to 180 with Seek 165

Move from 180 to 52 with Seek 128

Move from 52 to 43 with Seek 9

Move from 43 to 14 with Seek 29

Move from 14 to 68 with Seek 54

Move from 68 to 32 with Seek 36

Move from 32 to 25 with Seek 7

Move from 25 to 30 with Seek 5

Move from 30 to 54 with Seek 24

Total Seek Time is 492

Average Seek Time is 49.200001