**EXP NO: 11 PAGE REPLACEMENT ALGORITHM(FIFO)**

**AIM:**

To write a c program to implement page replacement algorithm(first in first out).

**ALGORITHM**:

1.Start th**e** program.

2.Read the number of frames.

3.Read the number of pages using for loop.

4.Initialize the values in frames to -1.

5.Check for condition frame[j]=page[i]. If it is true, then assign flag==1 or flag==0 and

Increment the top value for each frame frame[top]=page[i].

6.Allocate the pages in to frames in first in first out order.

7.Display the number of page faults.

8.Stop the program.

**CODING**:

#include< stdio.h>

#include<conio.h>

#define max 25

void main()

{

int frame[10];

int i,j,k,nf,np=0,page[max],temp;

int flag=0,pf=0,top=0;

printf("Enter no. of Frames:");

scanf("%d",&nf);

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

frame[i]=-1;

printf("Enter pages (press -999 to exit):\n");

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

{

scanf("%d",&temp);

if(temp==-999) break;

page[i]=temp;

np++;

}

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

{

flag=0;

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

{

if(frame[j]==page[i])

{

printf("\n\t");

flag=1;break;

}

}

if(flag==0)

{

frame[top]=page[i];

top++;

printf("\nFault: ");

pf++;

if(top>=nf)

top=0;

}

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

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

}

printf("\nNumber of page faults is: %d ",pf);

getch();

}

**OUTPUT**:

Enter no. of Frames:3

Enter pages (press -999 to exit):

7

0

1

2

0

3

0

4

2

3

0

3

1

2

0

-999

Fault: 7 -1 -1

Fault: 7 0 -1

Fault: 7 0 1

Fault: 2 0 1

2 0 1

Fault: 2 3 1

Fault: 2 3 0

Fault: 4 3 0

Fault: 4 2 0

Fault: 4 2 3

Fault: 0 2 3

0 2 3

Fault: 0 1 3

Fault: 0 1 2

0 1 2

Number of page faults is: 12

**RESULT:**

Thus the c program is executed for the implementation of first in first out page replacement algorithm and the output is verified.

**2.PAGE REPLACEMENT ALGORITHM(LRU)**

**AIM:**

To write a c program to implement least recently used page replacement algorithm.

:

1.Start the program.

2.Read the number of frames.

3.Read the number of pages using for loop.

4.Read the reference string using for loop.

5.Initialize for loop and check condition i<no of frames. If it is true, then initialize the

Frame to -1.

6.Allocate the pages into the frame by selecting the page that has not been used for the

Longest period of time.

7.Display the number of page faults.

8.Stop the program.

**CODING**:

#include<stdio.h>

int findLRU(int time[], int n){

int i, minimum = time[0], pos = 0;

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

if(time[i] < minimum){

minimum = time[i];

pos = i;

}

}

return pos;

}

int main()

{

int no\_of\_frames, no\_of\_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;

printf("Enter number of frames: ");

scanf("%d", &no\_of\_frames);

printf("Enter number of pages: ");

scanf("%d", &no\_of\_pages);

printf("Enter reference string: ");

for(i = 0; i < no\_of\_pages; ++i){

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

}

for(i = 0; i < no\_of\_frames; ++i){

frames[i] = -1;

}

for(i = 0; i < no\_of\_pages; ++i){

flag1 = flag2 = 0;

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == pages[i]){

counter++;

time[j] = counter;

flag1 = flag2 = 1;

break;

}

}

if(flag1 == 0){

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == -1){

counter++;

faults++;

frames[j] = pages[i];

time[j] = counter;

flag2 = 1;

break;

}

}

}

if(flag2 == 0){

pos = findLRU(time, no\_of\_frames);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

printf("\n");

for(j = 0; j < no\_of\_frames; ++j){

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

}

}

printf("\n\nTotal Page Faults = %d", faults);

return 0;

}

**L**

Enter number of frames: 3

Enter number of pages: 6

Enter reference string: 5

7

5

6

7

3

5 -1 -1

5 7 -1

5 7 -1

5 7 6

5 7 6

3 7 6

Total Page Faults = 4

**RESULT:**

Thus the c program is executed for the implementation of least recently used page replacement algorithm and the output is verified.

**3.PAGE REPLACEMENT ALGORITHM (OPTIMAL)**

**AIM:**

To write a c program to implement the optimal page replacement algorithm.

**ALGORITHM**

1.Start the program.

2.Enter the no of pages, no of frames, page reference number for each page.

3.Increase the fault value if the referred page is not found in the frames.

4.If a page is not present and a frame is empty fill the frame with the page.

5.If a page is not present find a page in the frame that will never be referenced in the

Future.

6.If no such page exists, find a page that is referenced farthest in future.

7.Replace this page with new page.

8.Display the page faults.

9.Stop the program.

**CODING:**

#include<stdio.h>

int main()

{

int no\_of\_frames, no\_of\_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k, pos, max, faults = 0;

printf("Enter number of frames: ");

scanf("%d", &no\_of\_frames);

printf("Enter number of pages: ");

scanf("%d", &no\_of\_pages);

printf("Enter page reference string: ");

for(i = 0; i < no\_of\_pages; ++i){

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

}

for(i = 0; i < no\_of\_frames; ++i){

frames[i] = -1;

}

for(i = 0; i < no\_of\_pages; ++i){

flag1 = flag2 = 0;

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == pages[i]){

flag1 = flag2 = 1;

break;

}

}

if(flag1 == 0){

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == -1){

faults++;

frames[j] = pages[i];

flag2 = 1;

break;

}

}

}

if(flag2 == 0){

flag3 =0;

for(j = 0; j < no\_of\_frames; ++j){

temp[j] = -1;

for(k = i + 1; k < no\_of\_pages; ++k){

if(frames[j] == pages[k]){

temp[j] = k;

break;

}

}

}

for(j = 0; j < no\_of\_frames; ++j){

if(temp[j] == -1){

pos = j;

flag3 = 1;

break;

}

}

if(flag3 ==0){

max = temp[0];

pos = 0;

for(j = 1; j < no\_of\_frames; ++j){

if(temp[j] > max){

max = temp[j];

pos = j;

}

}

}

frames[pos] = pages[i];

faults++;

}

printf("\n");

for(j = 0; j < no\_of\_frames; ++j){

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

}

}

printf("\n\nTotal Page Faults = %d", faults);

return 0;

}

**OUTPUT:**

Enter number of frames: 3

Enter number of pages: 10

Enter page reference string: 2

3

4

2

1

3

7

5

4

3

2 -1 -1

2 3 -1

2 3 4

2 3 4

1 3 4

1 3 4

7 3 4

5 3 4

5 3 4

5 3 4

Total Page Faults = 6

**RESULT:**

Thus the c program is executed for the implementation of optimal page replacement algorithm and the output is verified successfully.