**Practical No 12**

**Aim :** Develop, debug and Execute a C program to simulate LFU page replacement algorithms

**Apparatus:** Mingw compiler for C/C++, and a text editor for developing C code file (Dev C++).

**Theory :**

**What is Page Replacement Algorithm?**

* In operating systems that use paging for memory management, page replacement algorithms are needed to decide which page needed to be replaced when new page comes in.
* Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page.
* Different page replacement algorithms suggest different ways to decide which page to replace.
* The target for all algorithms is to reduce number of page faults.

**What is LFU Page Replacement?**

* LFU is short for Least Frequently Used page replacement Algorithm.
* In LFU algorithm the least frequently used page is removed and replaced with the new page
* LFU is one such page replacement policy in which the least frequently used pages are replaced.
* If the frequency of pages is the same, then the page that has arrived first is replaced first.

**What are the rules for LFU Page Replacement?**

**Example:**

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No of frames: 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F4 |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| F3 |  |  | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| F2 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F1 | 7 | 7 | 7 | 7 | 7 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 7 | 7 | 7 |
|  | \* | \* | \* | \* | HIT | \* | HIT | \* | HIT | HIT | HIT | HIT | HIT | \* | HIT | HIT | HIT | \* | HIT | HIT |

Page fault (\*): 8

Page hit (HIT): 12

Page fault ratio = No. of page fault / No. of reference string

= 8/20

= 40%

Page hit ratio = No. of page ratio / No. of reference string

= 12/20

= 60%

**Code:**

#include<stdio.h>

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

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

for(int 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;

}

printf("\n");

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

printf("F%d\t",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 = LRU(time, no\_of\_frames);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

printf("\n");

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

if(frames[j] == -1)

printf(" \t");

else

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

}

}

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

printf("\nTotal Page Hits = %d", no\_of\_pages - faults);

printf("\nPage Fault ratio = %.2f%%",faults/(float)no\_of\_pages \* 100);

printf("\nPage Hit ratio = %.2f%%",(no\_of\_pages - faults)/(float)no\_of\_pages \* 100);

return 0;

}

/\*

4

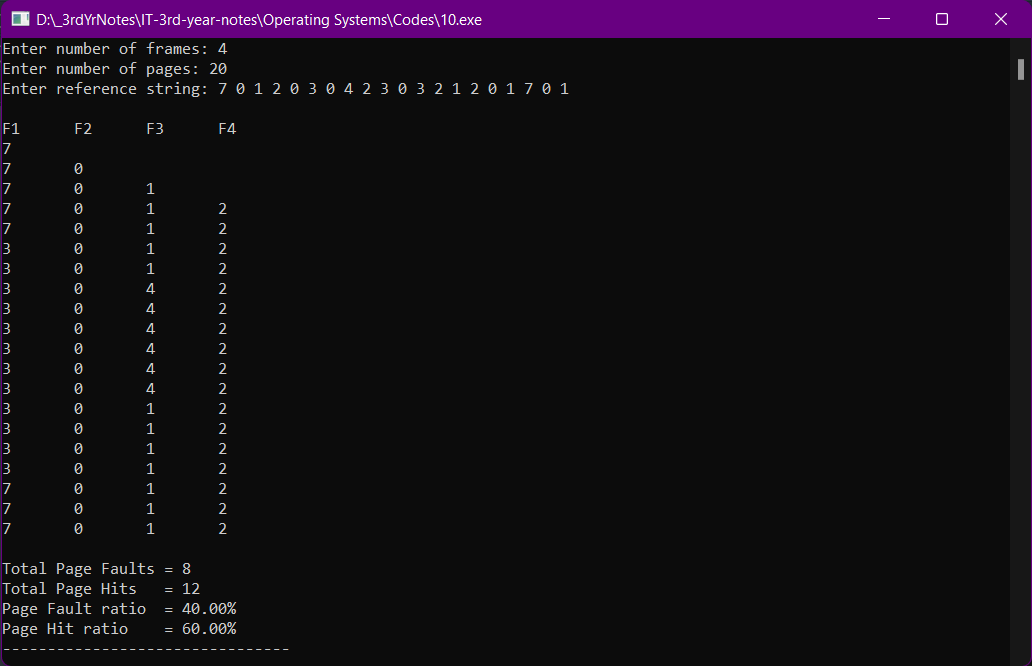
20

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Sample Test case

\*/

**Output:**



Output 10.1

**Conclusion**:

Hence, by performing this practical I got to know about the LRU page replacement algorithm i.e. Least Recently Used Page Replacement Algorithm. I also developed, debugged and executed a C program to simulate LRU page replacement algorithm.