**Practical No 11**

**Aim :** Develop, debug and Execute a C program to simulate Optimal 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 an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.
* The page replacement algorithms do this task of deciding which page needs to be replaced when a new page arrives in the memory.

**What are the components of Page Replacement Algorithms?**

**Page fault:**

* A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but is not loaded in physical memory.
* Since actual physical memory is much smaller than virtual memory, page faults can happen.
* In case of page faults, the operating system might have to replace one of the existing pages with the newly needed page.
* Different page replacement algorithms suggest different ways to decide which page to replace.
* The target for all algorithms is to reduce the number of page faults.

**Page Hit:**

* When we want to load the page on the memory, and the page is already available on memory, then it is called page hit.

**What is Optimal Page Replacement?**

* The idea is simple, for every reference we do following : If referred page is already present, increment hit count.
* If not present, find if a page that is never referenced in future. If such a page exists, replace this page with new page. If no such page exists, find a page that is referenced farthest in future. Replace this page with new page.

**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 main()

{

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

int isfault = 0, ishit = 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){

isfault = 0;

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++;

isfault = 1;

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");

int prevFrames[no\_of\_frames];

int hit\_miss\_flag[no\_of\_frames] ;

int sum = 0;

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

if(frames[j] == -1)

printf("\t ");

else

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

}

}

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

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

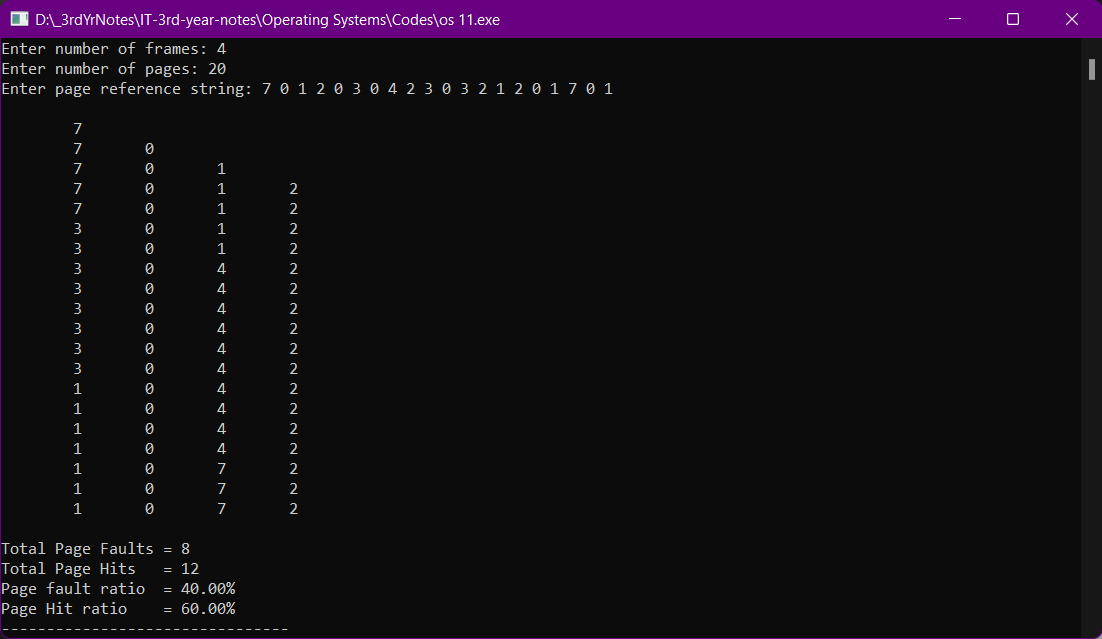
printf("\nPage fault ratio = %0.2f%%",faults/(float)no\_of\_pages \* 100);

printf("\nPage Hit ratio = %0.2f%%",(no\_of\_pages-faults)/(float)no\_of\_pages \* 100);

return 0;

}

**Output:**



**Conclusion**:

Hence, by performing this practical I got to know about the concept of Page replacement, page fault and page hit. I also learnt about optimal page replacement algorithm. I also developed, debugged and executed a C program to simulate optimal page replacement algorithm.