MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh,Tangail-1902

LAB REPORT

Lab Report No	: 11
Lab Report name	: Implementation of FIFO page replacement algorithm.
Course Title	: Operating System Lab
Course Code	: ICT-3110
Date of Performance	: 20-08-2020
Date of Submission	: 28-08-2020

Submitted by,

Student Name : Tanvir Ahmed

Student ID : IT-18043

Session : 2017-18

3rd Year 1st semester

Dept. of ICT

Submitted to,

Nazrul Islam

Assistant Professor

Dept. of ICT,

MBSTU.

Lab Report No. **11**

Lab Report Name: **Implementation of FIFO page replacement algorithm .**

Objectives:

i. What is FIFO page replacement algorithm.
ii. How to implementation

Theory:

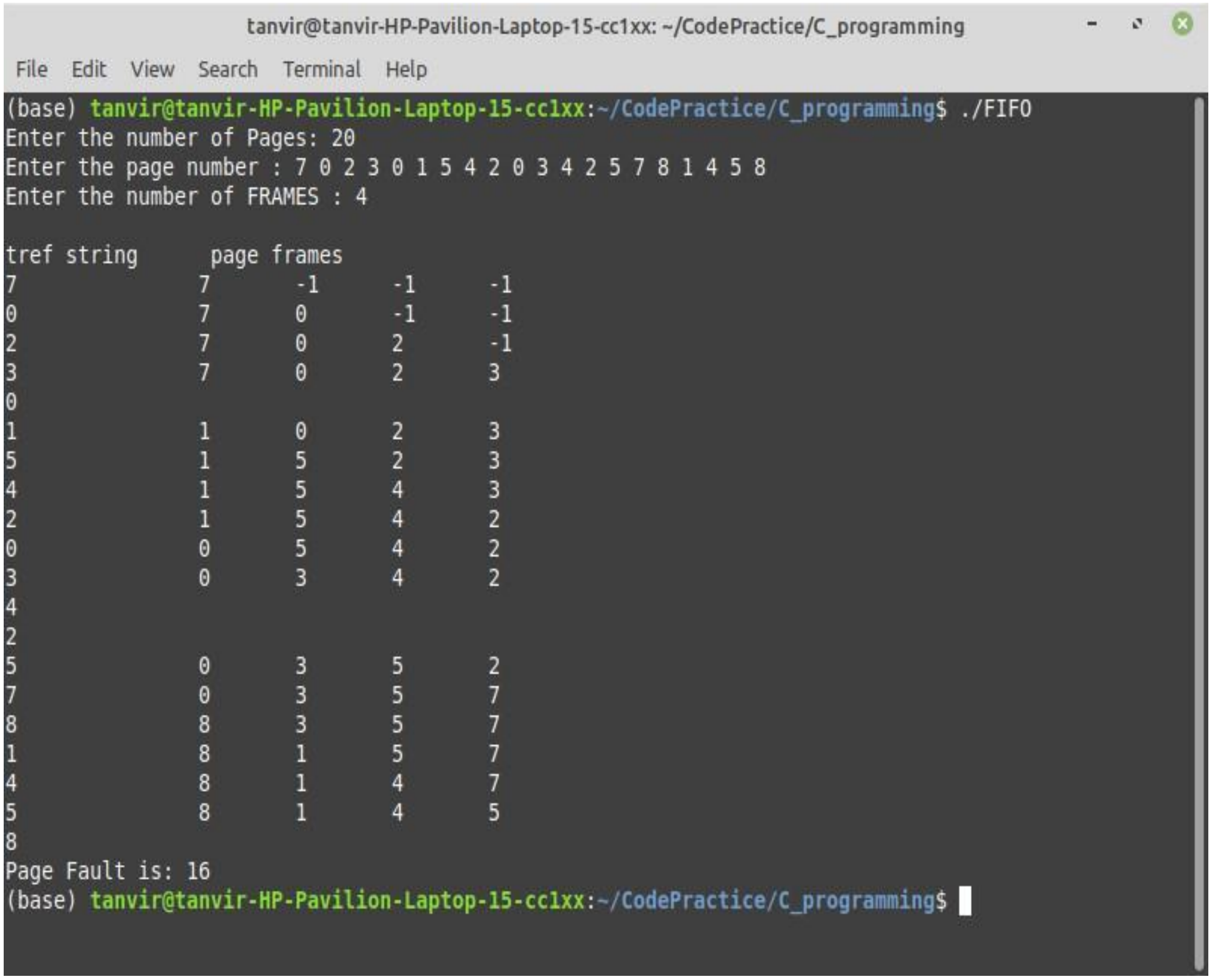
This is the simplest page replacement algorithm. In a page replacement algorithm we decide when a page replacement occures then which frames are to be replaced. For evaluating an algorithm we take a particular string of memory references ,called reference string.

In FIFO page replacement algorithm- for each page we track the time when it was brought into the memory and when any replacement request comes then oldest page is chosen. If we choose a queue to hold all pages in memory then its more easy to understand and implement rather than tracking time of all pages.

Corresponding Code:

```
#include<stdio.h>

int main()

{

   int i,j,n,a[50],frame[10],no,k,avail,count=0;

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

   scanf("%d",&n);

   printf("Enter the page number : ");

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

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

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

   scanf("%d",&no);
   for(i=0; i<no; i++)
      frame[i]= -1;
   j=0;
   printf("\n");
   printf("tref string\t page frames\n");

   for(i=1; i<=n; i++)
   {
      printf("%d\t\t",a[i]);
      avail=0;
      for(k=0; k<no; k++)
          if(frame[k]==a[i])
            avail=1;
      if (avail==0)
      {
         frame[j]=a[i];
         j=(j+1)%no;
         count++;
         for(k=0; k<no; k++)
            printf("%d\t",frame[k]);
      }
      printf("\n");
   }
   printf("Page Fault is: %d",count);
   printf("\n");
   return 0;
}
```

Output:


tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx: ~/CodePractice/C_programming

```
(base) tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx:~/CodePractice/C_programming$ ./FIFO
Enter the number of Pages: 20
Enter the page number : 7 0 2 3 0 1 5 4 2 0 3 4 2 5 7 8 1 4 5 8
Enter the number of FRAMES : 4

tref string      page frames
7               7       -1      -1      -1
0               7       0       -1      -1
2               7       0       2       -1
3               7       0       2       3
0
1               1       0       2       3
5               1       5       2       3
4               1       5       4       3
2               1       5       4       2
0               0       5       4       2
3               0       3       4       2
4
2
5               0       3       5       2
7               0       3       5       7
8               8       3       5       7
1               8       1       5       7
4               8       1       4       7
5               8       1       4       5
8
Page Fault is: 16
(base) tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx:~/CodePractice/C_programming$
```

Conclusion: This algorithm is used for reducing page faults. A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Here we got 20 pages with 4 frames. By the FIFO page replacement method first all the frames are booked by new pages. After that the 0 page number came which was their. So no faults. That is the page hit. Then a new page 1 came. This was not there. So this is a page miss or page fault. So the oldest page 7 will be removed from the frame. Then this procedure will repeat again and again. That’s how FIFO page replacement works.

In this time we get page fault 16. That is not bad for 20 pages. But it can decrease in real life programs .

FIFO is easy to understand. It is very easy to implement. But not always good at performance. It may replace an active page to bring a new one and thus may cause a page fault of that page immediately. Another unexpected side effect is the FIFO anomaly or Belady's anomaly. This anomaly says that the page fault rate may increase as the number of allocated page frames increases.