Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 11

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance:

Date of Submission:

Submitted by

Name: Rafiul Hasan

ID: IT-18016

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment no : 11

Experiment Name Implementation of FIFO Page Replacement Algorithm.

Objective:- The objective of this lab report is to become familiar with the virtual memory address 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 not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, 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.

First In First Out (FIFO) –This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

Example-1: Consider page reference string 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1 with 3 page frames. Find number of page faults.

*=unchanged

0	2	1	6	4	0	1	0^{*}	3	1^{*}	2	1^{*}
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3
		1	1	1	0	0	0	0	0	2	2

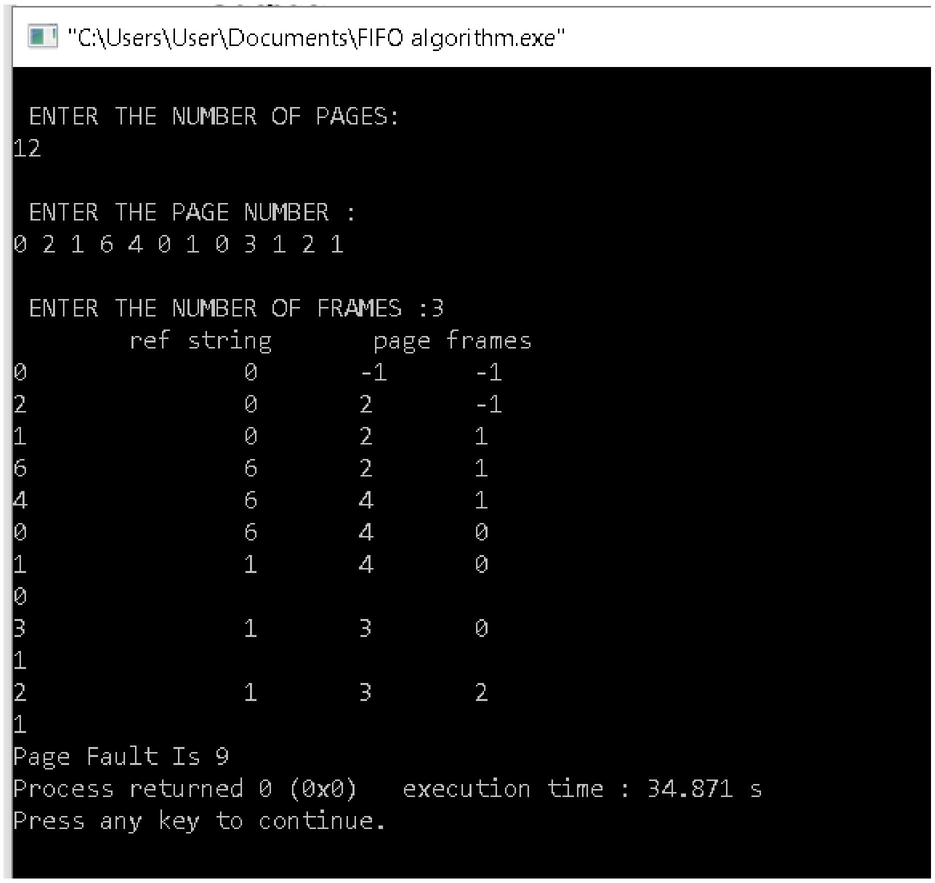
Page Fault = Total page –unchanged =12 – 3 = 9

Code:-

```
#include<stdio.h>

int main()

{

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

```
printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
printf("\n ENTER THE PAGE NUMBER :\n");
for(i=1; i<=n; i++)
   scanf("%d",&a[i]);
printf("\n ENTER THE NUMBER OF FRAMES :");
scanf("%d",&no);
for(i=0; i<no; i++)
   frame[i]=-1 ;
j=0;
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);
   return 0;
}
```

Output:



```
"C:\Users\User\Documents\FIFO algorithm.exe"

 ENTER THE NUMBER OF PAGES:
12

 ENTER THE PAGE NUMBER :
0 2 1 6 4 0 1 0 3 1 2 1

 ENTER THE NUMBER OF FRAMES :3
        ref string       page frames
0               0       -1      -1
2               0       2       -1
1               0       2       1
6               6       2       1
4               6       4       1
0               6       4       0
1               1       4       0
0
3               1       3       0
1
2               1       3       2
1
Page Fault Is 9
Process returned 0 (0x0)   execution time : 34.871 s
Press any key to continue.
```

Conclusion:- In the above study, we have found that optimal page replacement algorithm results the best algorithm because the average page faults in all the three cases with page frame size 2, 3 and 4 is less as compared to FIFO and LRU. A good page replacement algorithm reduces the number of page faults. In FIFO algorithm, some reference strings produces unexpected results called Belady's anomaly i.e. by increasing the number of page frames, the page fault also increases. In that case LRU works better than FIFO. In this paper, three different page replacement algorithms are studied and implemented with C# and compare with respect to page faults.