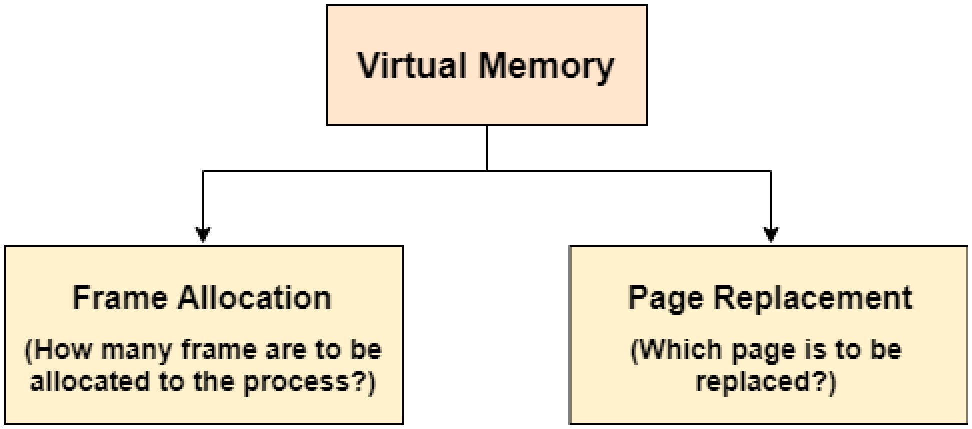
Lab Report No : 11

Lab Report name : Implementation FIFO page Replacement Algorithm .

Theory:

Page Replacement Algorithms

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).


Virtual Memory
Frame Allocation
(How many frame are to be allocated to the process?)
Page Replacement
(Which page is to be replaced?)

There are two main aspects of virtual memory, Frame allocation and Page Replacement. It is very important to have the optimal frame allocation and page replacement algorithm. Frame allocation is all about how many frames are to be allocated to the process while the page replacement is all about determining the page number which needs to be replaced in order to make space for the requested page.

Types of Page Replacement Algorithms

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

1. Optimal Page Replacement algorithm → this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.

2. Least recent used (LRU) page replacement algorithm → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.
3. FIFO → **i**n this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

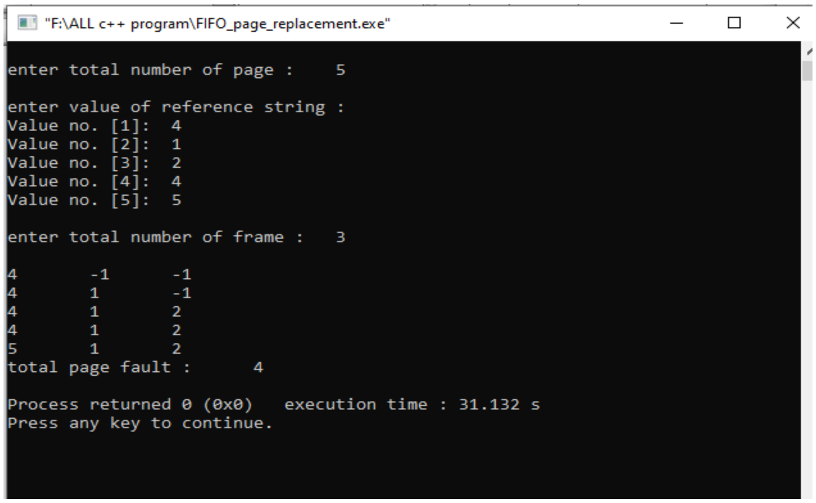
Implementaion FIFO page Replacement algorithm in c.

```
#include<stdio.h>
int main()
{
  int i, j, s, pa, frm, ref_str[10], page_flt= 0;
  printf("\nenter total number of page : \t");
  scanf("%d", &pa);
  printf("\nenter value of reference string : \n");
  for(i = 0; i < pa; i++)
  {
    printf("Value no. [%d]:\t", i + 1);

    scanf("%d", &ref_str[i]);
  }
  printf("\nenter total number of frame : \t");
  {
    scanf("%d", &frm);
  }
  int temp[frm];

  for(i= 0; i < frm; i++)
```

```
{
    temp[i] = -1;
}
for(i = 0; i < pa; i++)
{
    s = 0;
    for(j = 0; j< frm; j++)
    {
        if(ref_str[i] == temp[j])
        {
            s++;

            page_flt--;
        }
    }
    page_flt++;

    if((page_flt <= frm) && (s == 0))
    {
        temp[i] = ref_str[i];
    }
    else if(s == 0)
    {
        temp[(page_flt- 1) % frm] = ref_str[i];
    }
    printf("\n");
    for(j = 0; j < frm; j++)
    {
        printf("%d\t", temp[j]);
    }
}
```

```
    printf("\ntotal page fault : \t%d\n", page_flt);

}
```

Output:



```
enter total number of page :    5

enter value of reference string :
Value no. [1]:  4
Value no. [2]:  1
Value no. [3]:  2
Value no. [4]:  4
Value no. [5]:  5

enter total number of frame :   3

4       -1      -1
4       1       -1
4       1       2
4       1       2
5       1       2
total page fault :      4

Process returned 0 (0x0)   execution time : 31.132 s
Press any key to continue.
```

Discussion :

These page replacement algorithms are used in operating systems that support virtual memory management.

FIFO Page Replacement technique is one of the simplest one to implement amongst other page replacement algorithms. It is a conservative algorithm.
It is a low-overhead algorithm that maintains a queue to keep a track of all the pages in a memory.
When a page needs to be replaced, the page at the FRONT of the Queue will be replaced. The FIFO page replacement technique is not implemented in operating systems nowadays.