ASSIGNMENT NO.: 06

AIM:

Implement the C program for Page Replacement Algorithms: FCFS, LRU, and Optimal for frame size as minimum three.

PREREQUISITE:

1. C Programming
2. Fundamentals of Data Structure

OBJECTIVE:

To study

1. Page Replacement
2. Methods and Algorithms for Page Replacement

THEORY:

Page Replacement:

A computer system has a limited amount of memory. Adding more memory physically is very costly. Therefore, most modern computers use a combination of both hardware and software to allow the computer to address more memory than the amount physically present on the system. This extra memory is actually called Virtual Memory.

Virtual Memory is a storage allocation scheme used by the Memory Management Unit (MMU) to compensate for the shortage of physical memory by transferring data from RAM to disk storage. It addresses secondary memory as though it is a part of the main memory. Virtual Memory makes the memory appear larger than actually present which helps in the execution of programs that are larger than the physical memory.

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).

In Virtual Memory Management, Page Replacement Algorithms play an important role. The main objective of all the **Page replacement policies** is to decrease the maximum number of **page faults**.

Page Fault – It is basically a memory error, and it occurs when the current programs attempt to access the memory page for mapping into virtual address space, but it is unable to load into the physical memory then this is referred to as 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.

Virtual Memory can be implemented using two methods:

- Paging
- Segmentation

Paging

Paging is a process of reading data from, and writing data to, the secondary storage. It is a memory management scheme that is used to retrieve processes from the secondary memory in the form of pages and store them in the primary memory. The main objective of paging is to divide each process in the form of pages of fixed size. These pages are stored in the main memory in frames. Pages of a process are only brought from the secondary memory to the main memory when they are needed.

When an executing process refers to a page, it is first searched in the main memory. If it is not present in the main memory, a page fault occurs.

Page Fault is the condition in which a running process refers to a page that is not loaded in the main memory.

In such a case, the OS has to bring the page from the secondary storage into the main memory. This may cause some pages in the main memory to be replaced due to limited storage. A Page Replacement Algorithm is required to decide which page needs to be replaced.

Basic Page Replacement Algorithm in OS

Page Replacement Algorithm decides which page to remove, also called swap out when a new page needs to be loaded into the main memory.

When the page that was selected for replacement was paged out, and referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

Page Replacement technique uses the following approach. If there is no free frame, then we will find the one that is not currently being used and then free it. A-frame can be freed by writing its content to swap space and then change the page table in order to indicate that the page is no longer in the memory.

1. First of all, find the location of the desired page on the disk.
2. Find a free Frame: a) If there is a free frame, then use it. b) If there is no free frame then make use of the page-replacement algorithm in order to select the victim frame. c) Then after that write the victim frame to the disk and then make the changes in the page table and frame table accordingly.
3. After that read the desired page into the newly freed frame and then change the page and frame tables.
4. Restart the process.

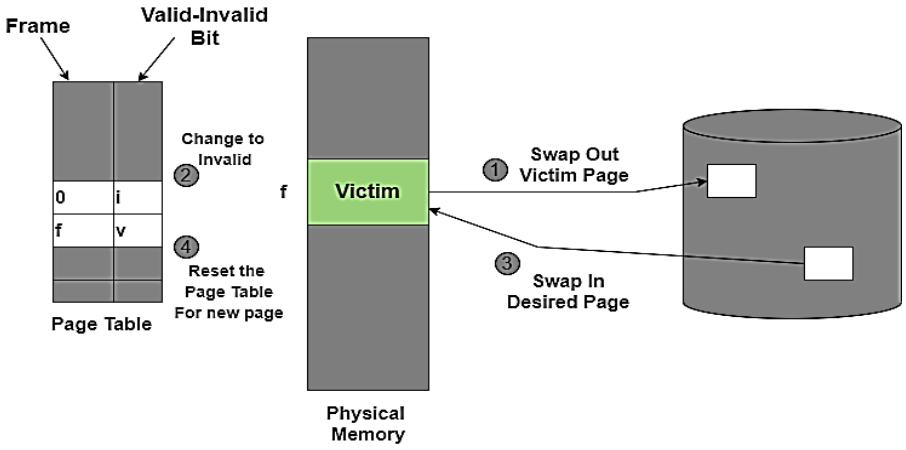
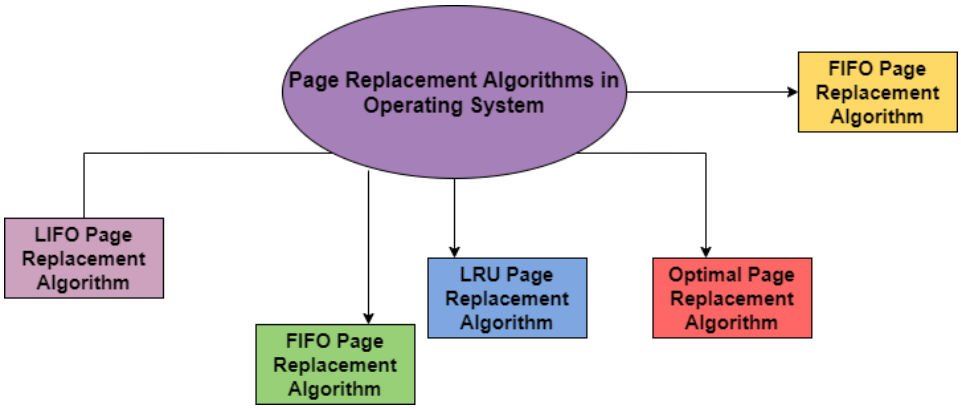


Figure: Page Replacement

Page Replacement Algorithms in OS

This algorithm helps to decide which pages must be swapped out from the main memory in order to create a room for the incoming page. This Algorithm wants the lowest page-fault rate.

Various Page Replacement algorithms used in the Operating system are as follows;



Some Page Replacement Algorithms:

1. First In First Out (FIFO)
2. Least Recently Used (LRU)
3. Optimal Page Replacement

1. First In First Out (FIFO)

It is a very simple way of Page replacement and is referred to as First in First Out. This algorithm mainly replaces the oldest page that has been present in the main memory for the longest time.

- This algorithm is implemented by keeping the track of all the pages in the queue.
- As new pages are requested and are swapped in, they are added to the tail of a queue and the page which is at the head becomes the victim.
- This is not an effective way of page replacement but it can be used for small systems.

Advantages

- This algorithm is simple and easy to use.
- FIFO does not cause more overhead.

Disadvantages

- This algorithm does not make the use of the frequency of last used time rather it just replaces the Oldest Page.

- There is an increase in page faults as page frames increases.
- The performance of this algorithm is the worst.

ALGORITHM

1. Start the process
2. Declare the size with respect to page length
3. Check the need of replacement from the page to memory
4. Check the need of replacement from old page to new page in memory
5. Forma queue to hold all pages
6. Insert the page require memory into the queue
7. Check for bad replacement and page fault
8. Get the number of processes to be inserted
9. Display the values
10. Stop the process

Code of FIFO Page Replacement Algorithm

```
#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:

```
ENTER THE NUMBER OF PAGES:  20
ENTER THE PAGE NUMBER :      7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
ENTER THE NUMBER OF FRAMES :3
     ref string      page frames
7          7    -1    -1
0          7    0     -1
1          7    0     1
2          2    0     1
0
3          2    3     1
0          2    3     0
4          4    3     0
2          4    2     0
3          4    2     3
0          0    2     3
3
2
1          0    1     3
2          0    1     2
0
1
7          7    1     2
0          7    0     2
1          7    0     1
Page Fault Is 15
```