Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 11

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 15/09/2020

Submitted by

Name: Md Ahadul Haque

ID:IT-18045

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.

Theory :

When a page fault occurs, the OS has to remove a page from the memory so that it can fit in another page in the memory.

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**.

Implementation:

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

Working Process :

Code for FIFO page replacement Algorithm –

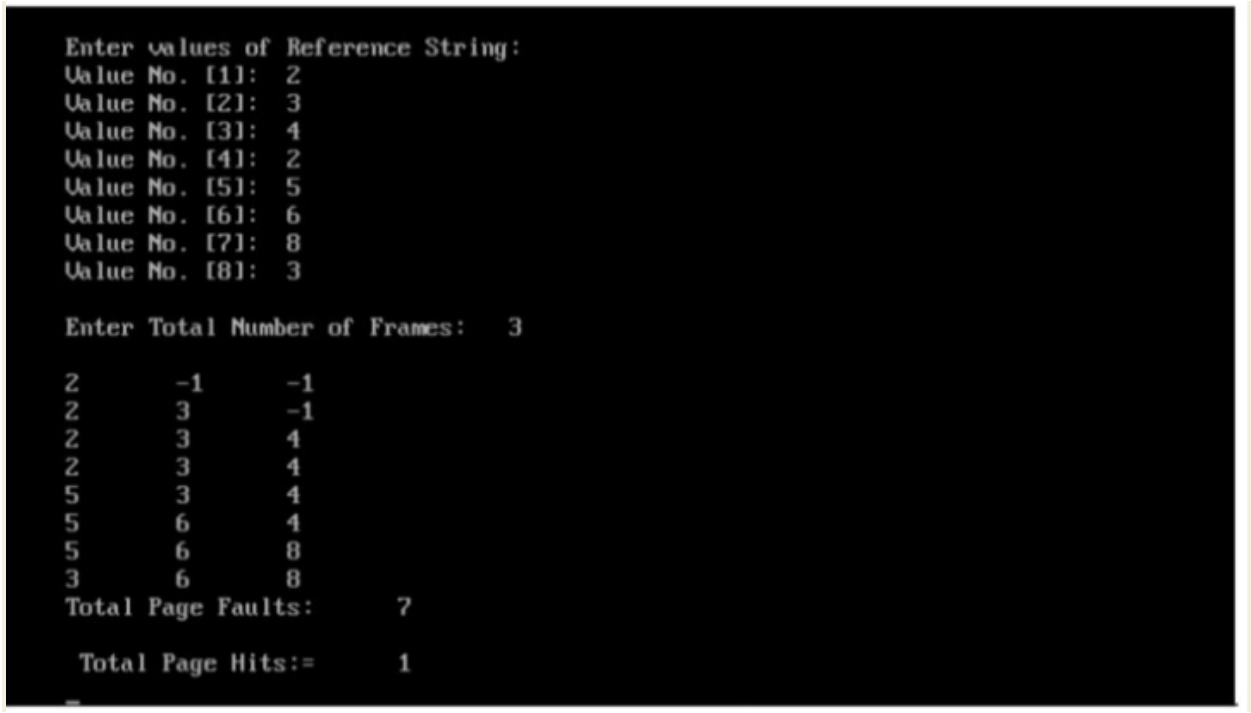
```
#include<stdio.h>
int main()
{
int reference_string[10],page_hits=0, page_faults = 0;
int temp[10],m, n, s, pages, frames;
clrscr();
printf("\n\n\t\t\t*******One Day Engineer*******\n\n");
printf("\nEnter Total Number of Pages:\t");
scanf("%d", &pages);
printf("\nEnter values of Reference String:\n");
for(m = 0; m < pages; m++)
{
printf("Value No. [%d]:\t", m + 1);
scanf("%d", &reference_string[m]);
}
printf("\nEnter Total Number of Frames:\t");
scanf("%d", &frames);
for(m = 0; m < frames; m++)
temp[m] = -1;
for(m = 0; m < pages; m++)
{
s = 0;
```

```
for(n = 0; n < frames; n++)
{
if(reference_string[m] == temp[n])
{
s++;
page_hits++;
page_faults--;
}
}
page_faults++;
if((page_faults <= frames) && (s == 0))
{
temp[m] = reference_string[m];
}
else if(s == 0)
{
temp[(page_faults - 1) % frames] = reference_string[m];
}
printf("\n");
for(n = 0; n < frames; n++)
printf("%d\t", temp[n]);
}

printf("\nTotal Page Faults:\t%d\n", page_faults);

printf("\n Total Page Hits:=\t%d\n",page_hits);
getch();
return 0;
}
```

Output:



Discussion :

It is simple and easy to understand & implement. The process effectiveness is low. When we increase the number of frames while using FIFO, we are giving more memory to processes . So, page fault should decrease, but here the page faults are increasing.Every frame needs to be taken account off.