Mawlana Bhashani Science and Technology University

Lab -Report

Report No:11
Course code: ICT-3110
Course title: Operating System Lab
Date of Performance:23-09-2020
Date of Submission:30-09-2020

Submitted by

Name: Ruku Shikder
ID: IT-18057
3th 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

- What is FIFO page replacement Algorithm?
- How to implement in C?

FIFO page replacement Algorithm:

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

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.

Algorithm:

Step 1: Start the program
Step 2: Read the number of frames

Step 3: Read the number of pages

Step 4: Read the page numbers

Step 5: Initialize the values in frames to -1

Step 6: Allocate the pages in to frames in First in first out order.

Step 7: Display the number of page faults.

Step 8: Stop the program

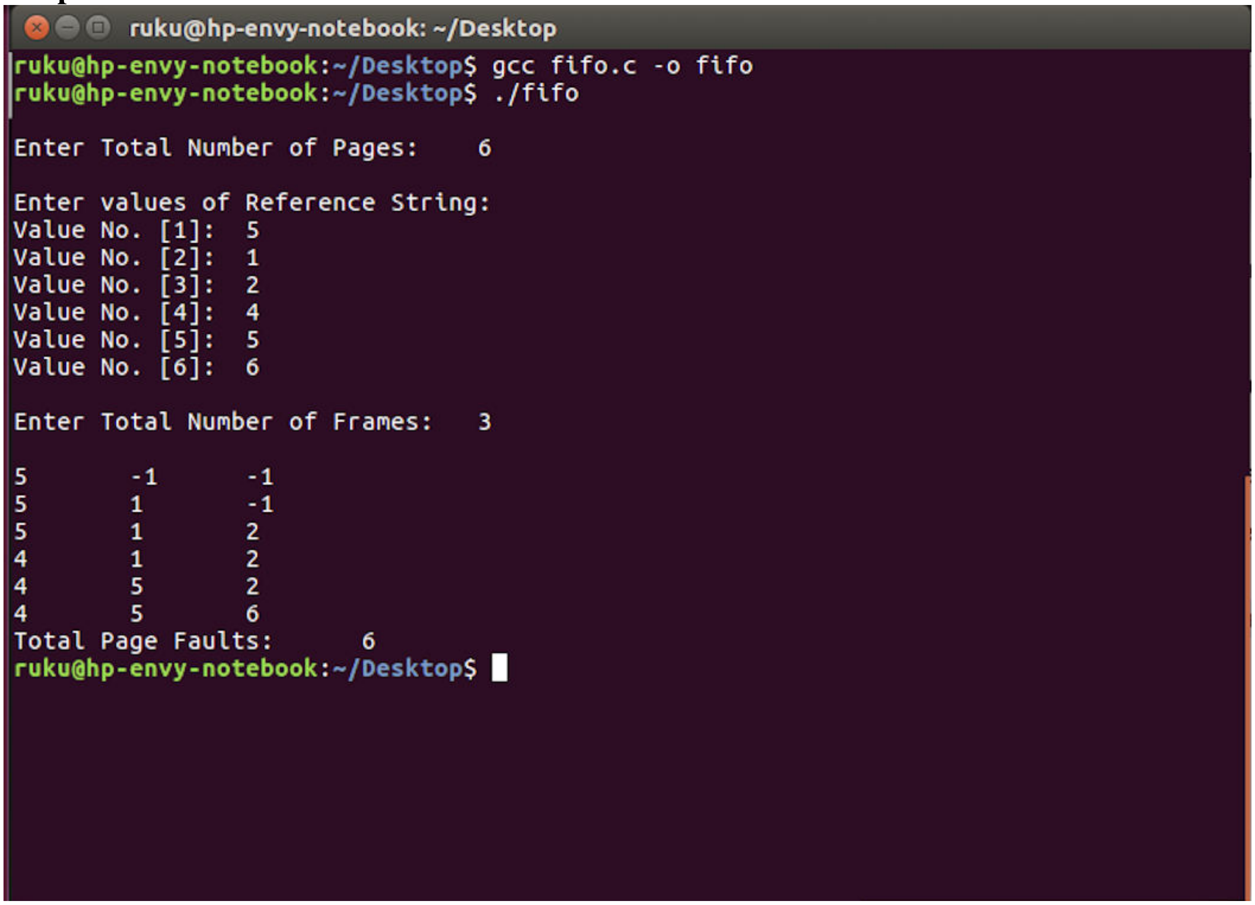
Implementation in C:
Code:

```
#include<stdio.h>

int main()
{
    int reference_string[10], page_faults = 0, m, n, s, pages, frames;
    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);
    }
    int temp[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_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);
      return 0;
}
```

Output:



ruku@hp-envy-notebook: ~/Desktop

```
ruku@hp-envy-notebook:~/Desktop$ gcc fifo.c -o fifo
ruku@hp-envy-notebook:~/Desktop$ ./fifo

Enter Total Number of Pages:    6

Enter values of Reference String:
Value No. [1]:  5
Value No. [2]:  1
Value No. [3]:  2
Value No. [4]:  4
Value No. [5]:  5
Value No. [6]:  6

Enter Total Number of Frames:   3

5       -1      -1
5       1       -1
5       1       2
4       1       2
4       5       2
4       5       6
Total Page Faults:      6
ruku@hp-envy-notebook:~/Desktop$ 
```

Discussion:

When the page that was selected for replacement and was paged out is referenced again it has to read in form disk and this requirds for I/O completion.This process determines the quality of the page replacement algorithm.The lesser thr time waiting for page-ins,the better is the algorithm.A page replacement algorithm looks at the limited information about accessing the pages provide by hardware and tires to selecte which pages should be replaced to minimize the total number of page misses while balancing it with the costs of primary storage and processor time of the algorithm itself.There are many different page replacement algorithms.We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults.