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: 12/09/2020

Submitted by

Name: Pritom Saha

ID: IT – 17010

3rd year 1st semester

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:

Page replacement algorithm is used to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and operating system replaces one of the existing pages with newly needed page. FIFO is a page replacement algorithm. FIFO stands for First In First Out. In this algorithm operating system keeps track of all pages in the memory in a queue, 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.

Working process:

1. **Run all the pages**
   - i) **If the size of the set is less than the capacity then**
     - a) **Insert page into the set one by one until the size of the set become equal to the capacity.**
     - b) **Also we have to maintain the pages in the queue to perform FIFO.**
     - c) **We have to increment page fault.**
   - ii) **Else**

     If current page is present in set, then do nothing.

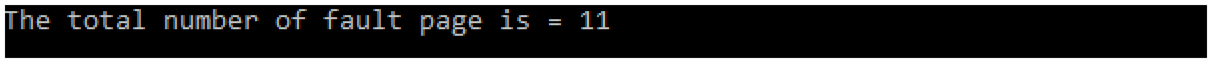
     Else
     - a) **We have to remove the first page from the queue.**
     - b) **Then we will replace the first page in the queue with the current page in the string.**
     - c) **Then we will store the current page in the queue.**
     - d) **At last we will increment page faults.**
2. **Return page faults.**

Code:

```
#include<bits/stdc++.h>

using namespace std;

unordered_set<int> unoset;

queue<int> idx;

int fault = 0;

signed main() {

        ios :: sync_with_stdio(false);

        cin.tie(0); cout.tie(0);

        int n = 15;

        int page[n] = {5, 3, 2, 0, 3, 8, 4, 5, 8, 10, 3, 5, 7, 3, 6};

        int cap = 5;

        int i = 0;

        do {

              i += 1;

              if (unoset.size() < cap) {

                    if (unoset.find(page[i]) == unoset.end()) {

                          unoset.insert(page[i]);

                          fault += 1;

                          idx.push(page[i]);

                    }

              } else {

                    if (unoset.find(page[i]) == unoset.end()) {
```

```
                    int value = idx.front();
                    idx.pop();
                    unoset.erase(value);
                    unoset.insert(page[i]);
                    idx.push(page[i]);
                    fault += 1;
                }
            }
        } while(i < n);
        cout << "The total number of fault page is = " << fault << endl;
        return 0;
}
```

Output:



```
The total number of fault page is = 11
```

Discussion:

To perform this algorithm we have to use STL. The time complexity of this algorithm is O (n log n) because various operations like find (), erase () works at O (log n) time complexity.