UNIVERSITY OF ASIA PACIFIC

Course Title: Operating System Lab
Course Code: CSE 406

Date of Submission: 10th October,2025

Submitted by:

Name: Md.Sarjil Hasan

Reg: 22101168

Sec : D

Submitted to:

Atia Rahman Orthi

Lecturer

University of Asia Pacific

Lab - 9:
FIFO Page Replacement Algorithm

Problem Description:

In operating systems, when a process executes, it needs to access pages stored in memory. However, the physical memory is limited, so not all pages of a process can be loaded at once.When a new page needs to be loaded and there is no free frame, a page replacement algorithm is used to decide which page to remove from memory.

The FIFO Page Replacement Algorithm is the simplest such technique. It replaces the oldest page in memory ,the one that has been in the system the longest –following the "first-in, first-out" principle.

Objective:

- To understand how the FIFO page replacement algorithm works.
- To implement FIFO page replacement using C++.
- To analyze the number of page faults generated by the algorithm.
- To compare its efficiency with other page replacement algorithms (like LRU or Optimal).

Algorithm Steps:

1. Initialize an empty queue to represent memory frames.
2. For each page reference in the reference string:
    - If the page is already in memory → no page fault.
    - Count each page fault.
    - If the page is not in memory:
        1. If there is still space → insert the page into the queue.
        2. If memory is full → remove the first (oldest) page from the queue and insert the new one.
3. After all references are processed, output the total number of page faults.

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, frames;
    cout << "Enter number of pages: ";
    cin >> n;
    vector<int> pages(n);
    cout << "Enter page reference string: ";
    for (int i = 0; i < n; i++)
        cin >> pages[i];

    cout << "Enter number of frames: ";
    cin >> frames;

    queue<int> q;
    unordered_set<int> s;
    int pageFaults = 0;

    cout << "\nPage\tFrames\t\tStatus\n";
    cout << "---------------------------------\n";
```

```cpp
for (int i = 0; i < n; i++) {
    int page = pages[i];
    bool fault = false;


    if (s.find(page) == s.end()) {
        fault = true;
        pageFaults++;
        if (s.size() == frames) {
            int oldest = q.front();
            q.pop();
            s.erase(oldest);
        }
        q.push(page);
        s.insert(page);
    }
    cout << page << "\t";
    vector<int> temp;
    queue<int> qtemp = q;
```

```
        while (!qtemp.empty()) {
            temp.push_back(qtemp.front());
            qtemp.pop();
        }
        for (int j = 0; j < temp.size(); j++)
            cout << temp[j] << " ";
        for (int j = temp.size(); j < frames; j++)
            cout << "- ";
        cout << "\t";

        if (fault)
            cout << "Page Fault";
        else
            cout << "Hit";

        cout << endl;
    }
    cout << "---------------------------------\n";
    cout << "Total Page Faults = " << pageFaults << endl;

    return 0;
}
```

Output:

```
Enter number of pages: 9
Enter page reference string: 3 2 0 1 0 5 9 1 2
Enter number of frames: 3

Page    Frames          Status
---------------------------------
3       3 - -   Page Fault
2       3 2 -   Page Fault
0       3 2 0   Page Fault
0       3 2 0   Page Fault
1       2 0 1   Page Fault
0       2 0 1   Hit
5       0 1 5   Page Fault
9       1 5 9   Page Fault
1       1 5 9   Hit
2       5 9 2   Page Fault
---------------------------------
Total Page Faults = 7
```

Discussion:

The FIFO page replacement algorithm is conceptually simple and easy to implement. It maintains pages in the order they entered memory and replaces the oldest one when needed. However, FIFO can suffer from Belady's Anomaly, where increasing the number of frames may actually increase the number of page faults — which makes it less efficient compared to more adaptive algorithms like LRU (Least Recently Used).In practical systems, FIFO is rarely used alone but serves as a foundational concept in understanding paging.

Advantages:

- Simple to understand and implement.
- Requires minimal computational overhead.
- Easy to simulate and analyze.

Disadvantages:

- Does not consider page usage frequency.
- May cause Belady's Anomaly.
- Can perform poorly for certain access patterns.

Conclusion:

The FIFO page replacement algorithm is an introductory and straightforward technique for page management. While easy to apply, it does not adapt well to changing reference patterns. Understanding FIFO provides a foundation for grasping more sophisticated algorithms like LRU and Optimal Page Replacement.

Git-Hub Link of my repository:

https://github.com/Sarjil-SarZzz/CSE406-LAB.git