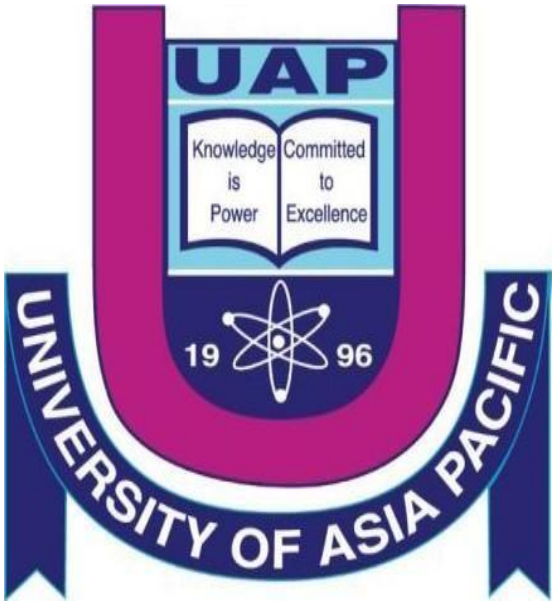

Department of Computer Science & Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Lab Report No: 10

Lab Report On, FIFO Page Replacement

Submitted To:
Atia Rahman Orthi
Lecturer,
Department of CSE, UAP

Submitted By:
Name: Amirul Islam Papon
Reg No: 21201076
Sec: B2

Problem Statement

In computer systems, memory management is a critical component of operating system design. When the number of pages requested by processes exceeds the number of available page frames in memory, a page replacement algorithm is required to decide which pages to remove to make space for new ones. The **FIFO (First-In, First-Out)** Page Replacement algorithm is one such method. The problem involves efficiently managing memory pages to minimize page faults while ensuring all required pages can be loaded and executed by processes.

Algorithm Steps (FIFO)

1. Initialize an empty memory queue with a fixed capacity (number of frames).
2. For each page request in the reference string:
    - If the page is already in memory, do nothing.
    - If the page is not in memory:
        - If the memory is not full, add the page to memory.
        - If the memory is full, remove the oldest page (the one that entered first) and insert the new one.
3. Count each insertion of a new page (that causes a removal or adds to a non-full memory) as a page fault.
4. Continue the process until all page references have been processed.

Code

fifo_page_replacement.py > ...

```python
def fifo_page_replacement(pages, capacity):
    memory = []
    page_faults = 0
    index = 0
    for page in pages:
        if page not in memory:
            if len(memory) < capacity:
                memory.append(page)
            else:
                memory[index] = page
                index = (index + 1) % capacity
            page_faults += 1
        print(f"Page: {page} -> Memory: {memory}")

    print(f"\nTotal Page Faults: {page_faults}")
    return page_faults

pages = [1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5]
capacity = 3
fifo_page_replacement(pages, capacity)

```

Output

```
[Running] python -u "d:\Operating-System\fifo_page_replacement.py"
Page: 1 -> Memory: [1]
Page: 2 -> Memory: [1, 2]
Page: 3 -> Memory: [1, 2, 3]
Page: 4 -> Memory: [4, 2, 3]
Page: 1 -> Memory: [4, 1, 3]
Page: 2 -> Memory: [4, 1, 2]
Page: 5 -> Memory: [5, 1, 2]
Page: 1 -> Memory: [5, 1, 2]
Page: 2 -> Memory: [5, 1, 2]
Page: 3 -> Memory: [5, 3, 2]
Page: 4 -> Memory: [5, 3, 4]
Page: 5 -> Memory: [5, 3, 4]

Total Page Faults: 9
```

Discussion

The FIFO page replacement algorithm is simple and easy to implement. It uses a queue data structure to keep track of the order in which pages were added to memory. However, the algorithm does not consider how often or how recently a page is used, which can lead to inefficient replacements.

Conclusion

The FIFO Page Replacement algorithm is a foundational concept in operating system memory management. While it's easy to implement and understand, it is not always efficient in minimizing page faults due to its simplistic approach. It can be a useful choice in systems where performance is not critical, or as a reference point when evaluating more advanced algorithms.

Source Code: https://github.com/Amirul-Islam-Papon/Operating-System/blob/main/fifo_page_replacement.py