

**Lab Report Of Operating System On**

***FCFS (Frist Come First Serve) and LRU PRE (Least Recently Used ) page replacement algorithm in Visual Studio using .NET(C#)***

Lab Report No: 05 & 06

**SUBMITTED BY: SUBMITTED TO:**

Name: Saurav Shrestha Name: Er. Pratap Sapkota

Semester : 4th Semester Signature: Matrix No: C30105220035

# FCFS PRE (First-Come-First-Serve Page Replacement {Lab 5})

It is a page replacement algorithm used in computer operating systems to manage memory. In FCFS PRE, the page that has been in memory the longest is the one selected for replacement. This algorithm follows the principle of "first come, first served," meaning the page that entered memory first is the first one to be replaced when a page fault occurs.

1. **LRU PRE (Least Recently Used Page Replacement {Lab 6})**

It is a widely-used page replacement algorithm in computer operating systems, particularly in memory management. It aims to minimize the number of page faults by evicting the page that has not been accessed for the longest time when a new page needs to be brought into memory**.**

## Question

## WAP to calculate perform FIFO PRE for the given scenario

**Page Queue**

1 3 7 2 1 4 3 3 2 1 6 0 4 1 2

calculate No. of page fault too. Using 3 page frames.

## Code

**FCFS (*Frist Come First Serve* )**

using System;

using System.Collections.Generic;

class MainClass

{

public static void Main(string[] args)

{

int[] pages = { 1, 3, 7, 2, 1, 4, 3, 3, 2, 1, 6, 0, 4, 1, 2 };

int pageSize = 3;

int pageFault = 0;

// Create a queue to represent the page frames

Queue<int> pageFrames = new Queue<int>(pageSize);

// Iterate through each page in the sequence

for (int i = 0; i < pages.Length; i++)

{

int currentPage = pages[i];

// Insert the first three pages without counting page faults

if (i < pageSize)

{

pageFrames.Enqueue(currentPage);

}

else

{

// Check if the page is already in memory

if (!pageFrames.Contains(currentPage))

{

// Page fault occurred

pageFault++;

// If the page frames are full, remove the oldest page

if (pageFrames.Count == pageSize)

{

pageFrames.Dequeue();

}

// Add the current page to the page frames

pageFrames.Enqueue(currentPage);

}

}

// Print the current state of the page frames

Console.Write($"Page {currentPage}: ");

foreach (int page in pageFrames)

{

Console.Write(page + " ");

}

Console.WriteLine();

}

// Print the total number of page faults

Console.WriteLine($"Total Page Faults: {pageFault}");

}

}

# Output

*Figure 1 FCFS*

**LRU PRE (Least Recently Used Page Replacement {Lab 6})**

# using System;

# using System.Collections.Generic;

# class MainClass

# {

# public static void Main(string[] args)

# {

# int[] pages = { 1, 3, 7, 2, 1, 4, 3, 3, 2, 1, 6, 0, 4, 1, 2 };

# int pageSize = 3;

# int pageFault = 0;

# // Create a queue to represent the page frames

# Queue<int> pageFrames = new Queue<int>(pageSize);

# // Create a dictionary to keep track of the last usage index of each page

# Dictionary<int, int> pageLastUsedIndex = new Dictionary<int, int>();

# // Track the number of pages encountered

# int pagesEncountered = 0;

# // Iterate through each page in the sequence

# for (int i = 0; i < pages.Length; i++)

# {

# int currentPage = pages[i];

# // Update the last usage index of the current page

# pageLastUsedIndex[currentPage] = i;

# // If the page is among the first three, just add it to page frames

# if (pagesEncountered < 3)

# {

# pageFrames.Enqueue(currentPage);

# pagesEncountered++;

# }

# else

# {

# // Check if the page is already in memory

# if (!pageFrames.Contains(currentPage))

# {

# // Page fault occurred

# pageFault++;

# // Remove the least recently used page

# int leastRecentlyUsedPage = pageFrames.Dequeue();

# pageLastUsedIndex.Remove(leastRecentlyUsedPage);

# // Add the current page to the page frames

# pageFrames.Enqueue(currentPage);

# }

# else

# {

# // If the page is already in memory, update its usage index

# pageFrames.Enqueue(pageFrames.Dequeue());

# }

# }

# // Print the current state of the page frames

# Console.Write($"Page {currentPage}: ");

# foreach (int page in pageFrames)

# {

# Console.Write(page + " ");

# }

# Console.WriteLine();

# }

# // Print the total number of page faults

# Console.WriteLine($"Total Page Faults: {pageFault}");

# }

# }

# Output

*Figure 1 LRU output*