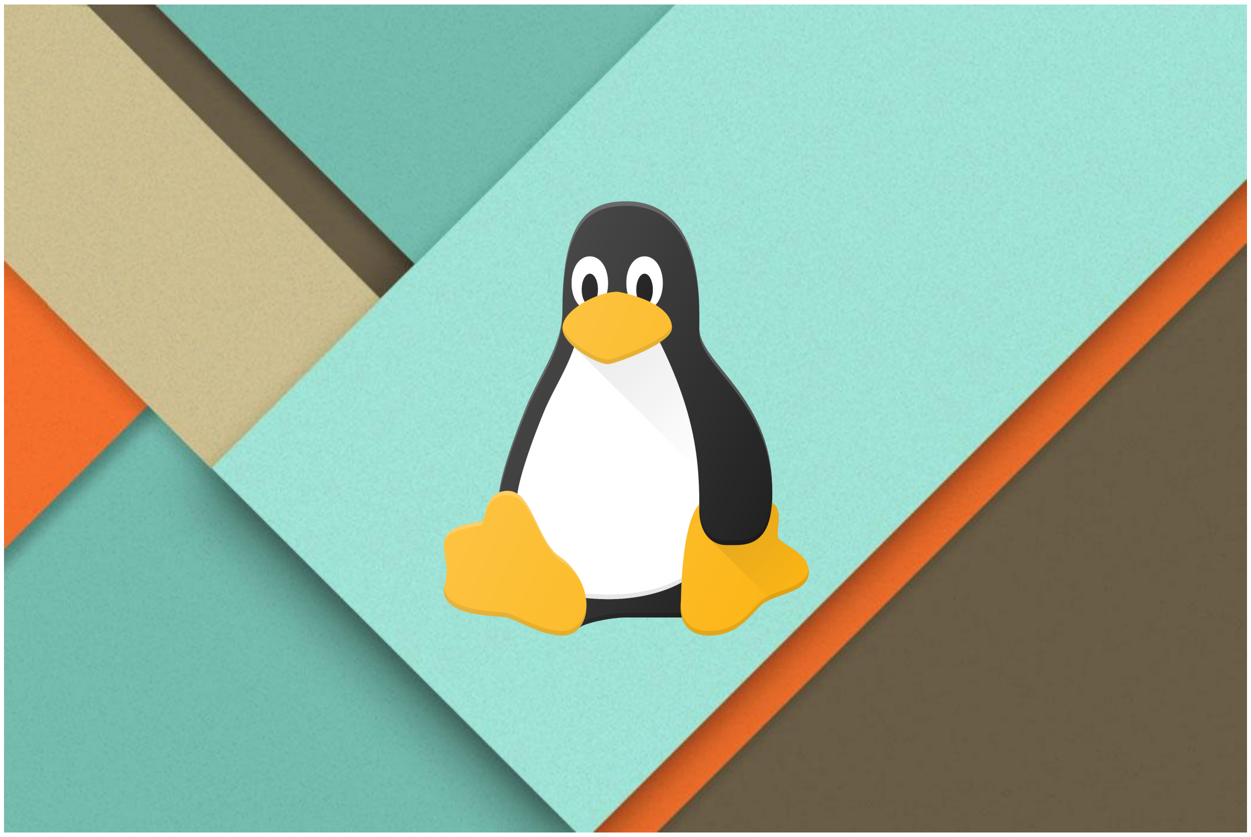


Operating Systems Lab

Experiment No. 9

8.04.2022

Professor - Dr. Shrinivas Khedkar.

Pratham Loya
201080068
IT
prloya_b20@it.vjti.ac.in

Aim

Write a program that implements the FIFO, LRU, and optimal page replacement algorithms. First, generate a random page-reference string where page numbers range from 0 to 9.

Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithms so that the number of page frames can vary from 1 to 7. Assume that demand paging is used.

Theory

In operating systems that use paging for memory management, page replacement algorithms are needed to decide which page needs to be replaced when a new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and the Operating System replaces one of the existing pages with a 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.

FIFO Page Replacement Algorithm

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, the page in the front of the queue is selected for removal.

Consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots.

Initially all slots are empty, so when 1, 3, 0 come they are allocated to the empty slots —> 3 **Page Faults.**

when 3 comes, it is already in memory so —> 0 **Page Faults.**

Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>1 **Page Fault.**

Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>1 **Page Fault.**

LRU Page Replacement Algorithm

The Least **R**ecently **U**sed (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used. The idea is based on locality of reference, the least recently used page is not likely

Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially we have 4 page slots empty.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already there so —> **0 Page fault.**

when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —> **0 Page fault**.

4 will takes place of 1 —> **1 Page Fault**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal Page Replacement Algorithm

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults

0 is already there so —> **0 Page fault**.

when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**

0 is already there so —> **0 Page fault.**

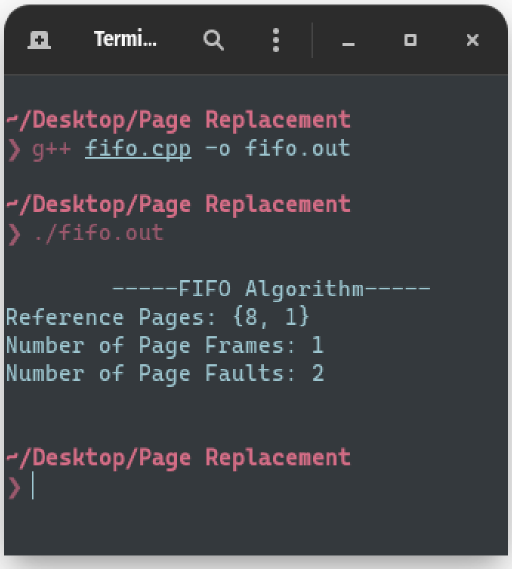
4 will takes place of 1 —> **1 Page Fault.**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Program

fifo.cpp



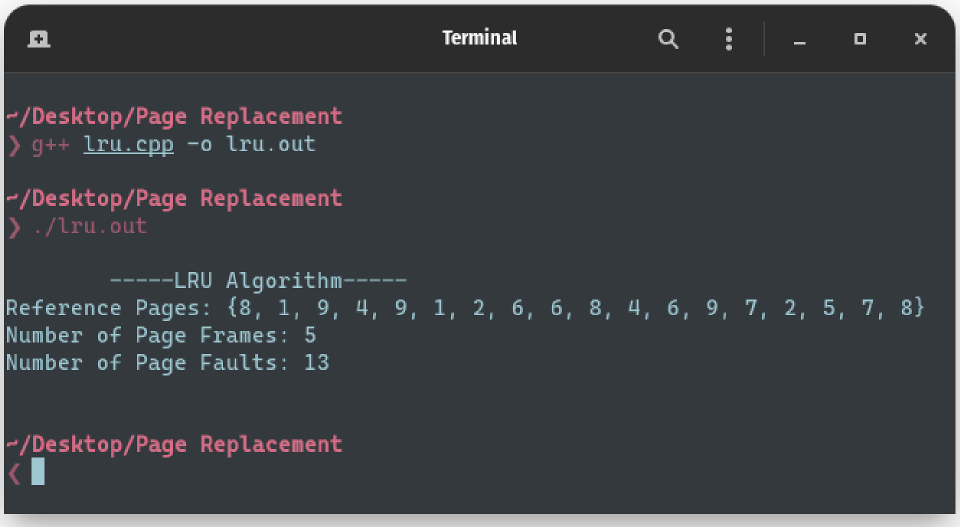
```
~/Desktop/Page Replacement
❯ g++ fifo.cpp -o fifo.out

~/Desktop/Page Replacement
❯ ./fifo.out

        -----FIFO Algorithm-----
Reference Pages: {8, 1}
Number of Page Frames: 1
Number of Page Faults: 2


~/Desktop/Page Replacement
❯ |
```

lru.cpp



Terminal

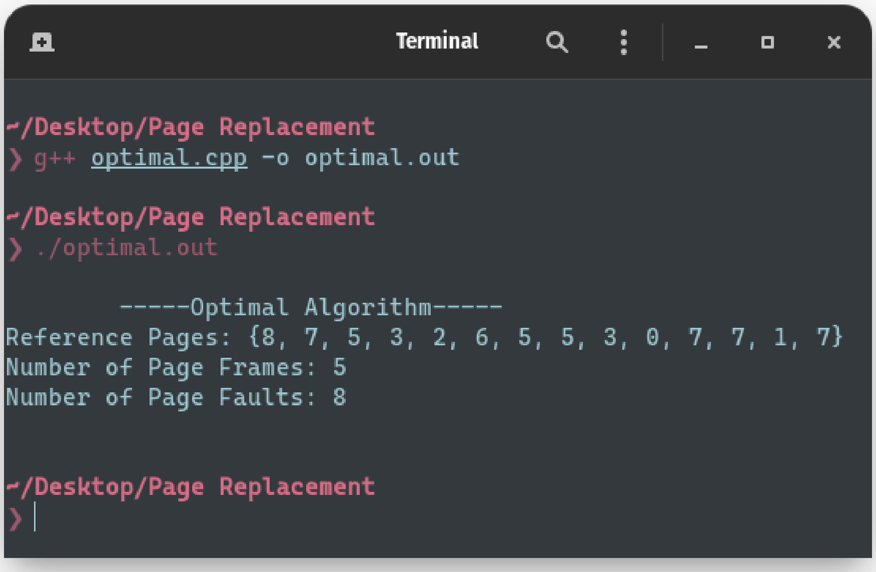
```
~/Desktop/Page Replacement
❯ g++ lru.cpp -o lru.out

~/Desktop/Page Replacement
❯ ./lru.out

        -----LRU Algorithm-----
Reference Pages: {8, 1, 9, 4, 9, 1, 2, 6, 6, 8, 4, 6, 9, 7, 2, 5, 7, 8}
Number of Page Frames: 5
Number of Page Faults: 13


~/Desktop/Page Replacement
❮
```

optimal.cpp



Terminal

```
~/Desktop/Page Replacement
❯ g++ optimal.cpp -o optimal.out

~/Desktop/Page Replacement
❯ ./optimal.out

        -----Optimal Algorithm-----
Reference Pages: {8, 7, 5, 3, 2, 6, 5, 5, 3, 0, 7, 7, 1, 7}
Number of Page Frames: 5
Number of Page Faults: 8


~/Desktop/Page Replacement
❯
```

Thank You