CSEN 383 – Assignment 4
Winter 2024

Group 2

Chaitra Boggaram (W1651213)
Divyanth Chalicham (07700005995)
Shreya Chinthala (07700005606)
Ruchi Manikrao Dhore (W1652116)
Rohit Parthiban (07700006502)

Objective

The objective of this project is to gain hands-on experience to explore the various memory management algorithms for swapping and paging. Using C program to operate various page replacement algorithms such as FIFO, LRU, LFU, MFU, and random pick. We have to generate random processes with varied sizes and durations, organizing them into a Job Queue based on arrival time. Additionally, we have to manage physical memory frames and disk pages, allocating memory to processes from the free pages list. Also, generate the appropriate record whenever starting or completing a job. We need to apply the chosen page replacement algorithm to select a victim page to evict so you can bring to memory the needed page. We have implemented various page replacement algorithms such as FIFO, LRU, LFU, MFU, and random pick. Hit and miss statistics are tracked for every run as well as calculate the average number of processes successfully swapped-in for each replacement algorithm over multiple runs and print.

Theory

In understanding the dynamics of page replacement algorithms, it's crucial to delve into the distinct strategies they employ. The FCFS Algorithm, standing for First-Come-First-Served, adheres strictly to the principle of chronology, opting to replace the page that arrived earliest. Conversely, the LRU Algorithm, which stands for Least Recently Used, operates on the principle of recency, choosing to replace the page that has remained untouched for the longest duration. The LFU Algorithm, denoting Least Frequently Used, prioritizes the replacement of pages with the least number of references, aiming to optimize memory usage by eliminating infrequently accessed pages. In contrast, the MFU Algorithm, or Most Frequently Used, targets pages that have been referenced most often, favoring those with higher utilization rates to maintain efficiency. Lastly, the R Algorithm introduces an element of randomness into the replacement process, selecting pages for eviction arbitrarily. Each of these algorithms offers a unique approach to managing memory resources, catering to various scenarios and optimizing system performance accordingly.

Presumptions

Each process does start at page-0 then every 100 msec it references a random page from its own address space.

- Locality of reference, after referencing a page i, there is a 70% probability that the next reference will be to page i, i-1, or i+1. It wraps around from 10 to 0. In other words, there is a 70% probability that for a given i, Δi will be -1, 0, or +1. Otherwise, $|\Delta i| > 1$.
- We will be generating 150 jobs.
- Sort the random jobs generation based on arrival time and have them structured as a linked list.
- Processes have randomly and evenly distributed sizes of 5, 11, 17, and 31 MB.
- Processes have randomly and evenly distributed service durations of 1, 2, 3, 4, or 5 seconds.

Code Execution

- A Makefile has been created to compile all the files and showcase the output. To execute the code, use the following command

```
make run
```

- To remove the object files and clean up the project directory, employ the command

```
make clean
```

Result & Conclusion

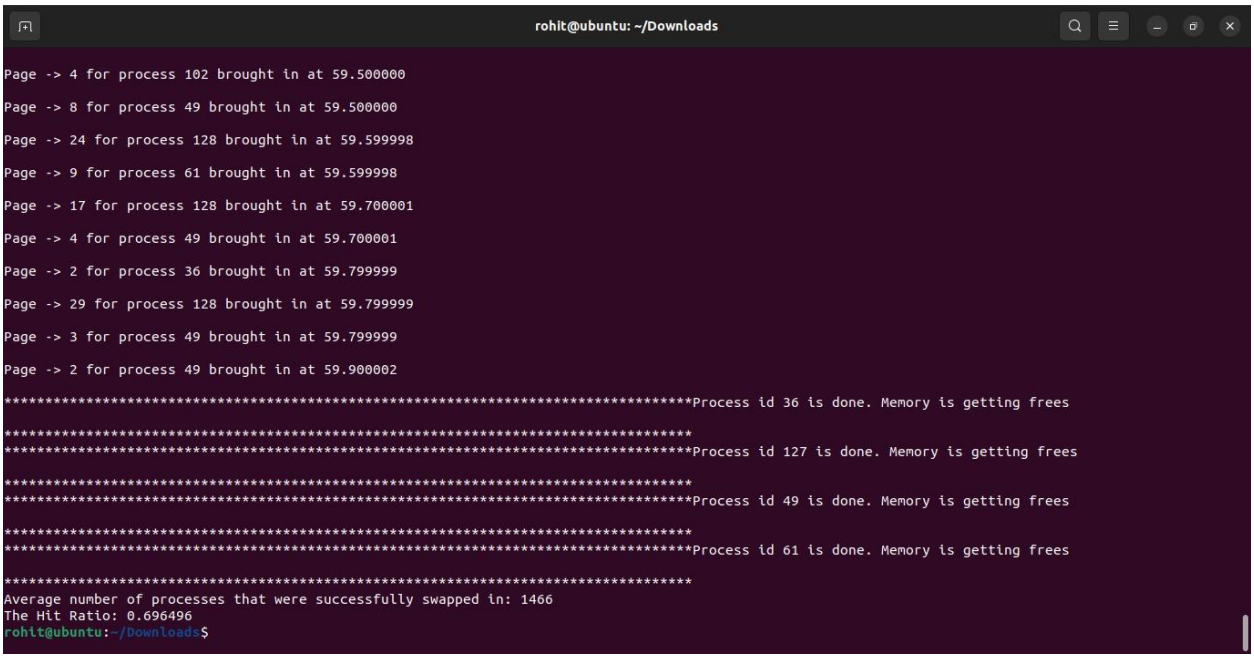
The Swapping and Paging Simulation project offers insightful information about how different page replacement methods impact an operating system's memory management and page swapping. Through a series of simulations using various methods, the code facilitates performance comparison and analysis. The best performance in terms of Hit Ratio - we get is Least Recently Used Algorithm.

Algorithm	Hit Ratio	Pages Swapped
FCFS	0.6964	1466
LFU	0.6561	1648
LRU	0.6935	1485
MFU	0.6701	1587
Random	0.6945	1517

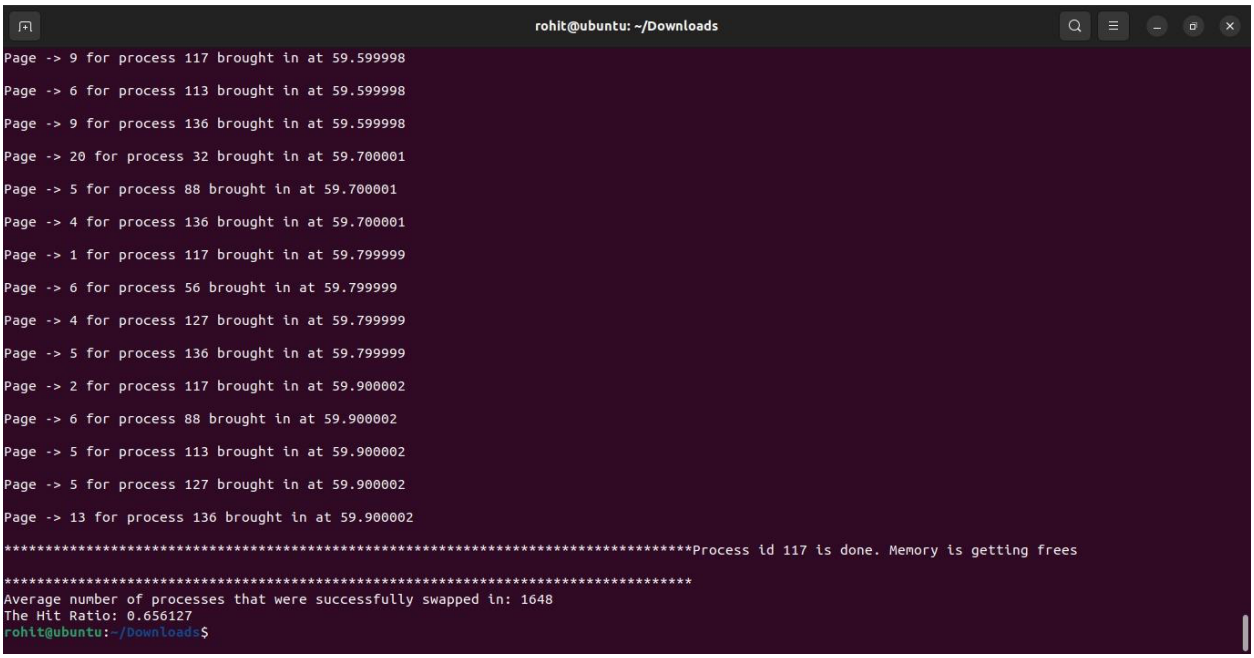
Outputs

Following are some screenshots from a run on Linux machine:

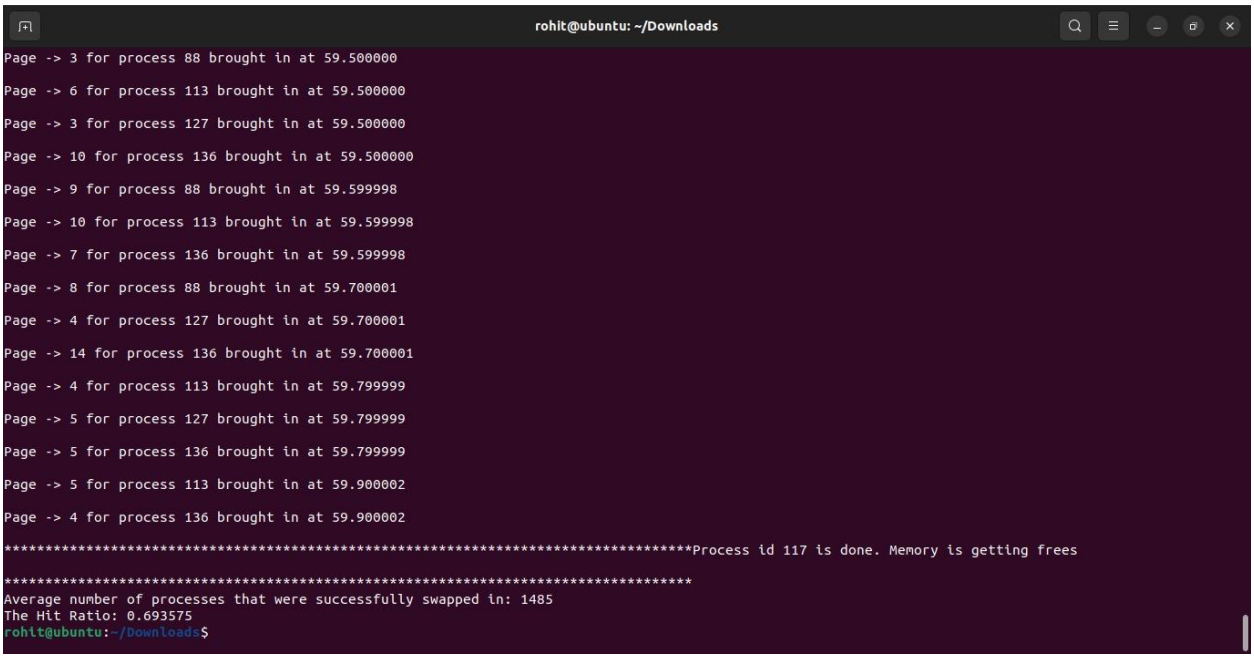
First Come First Serve



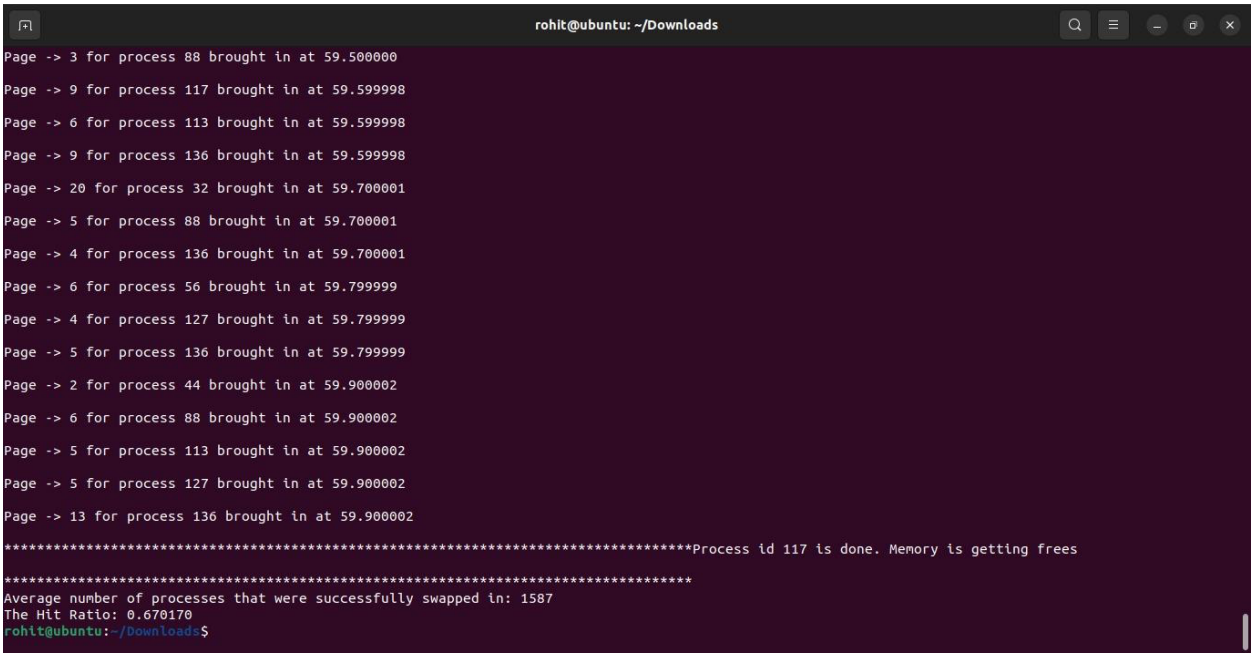
Least Frequently Used



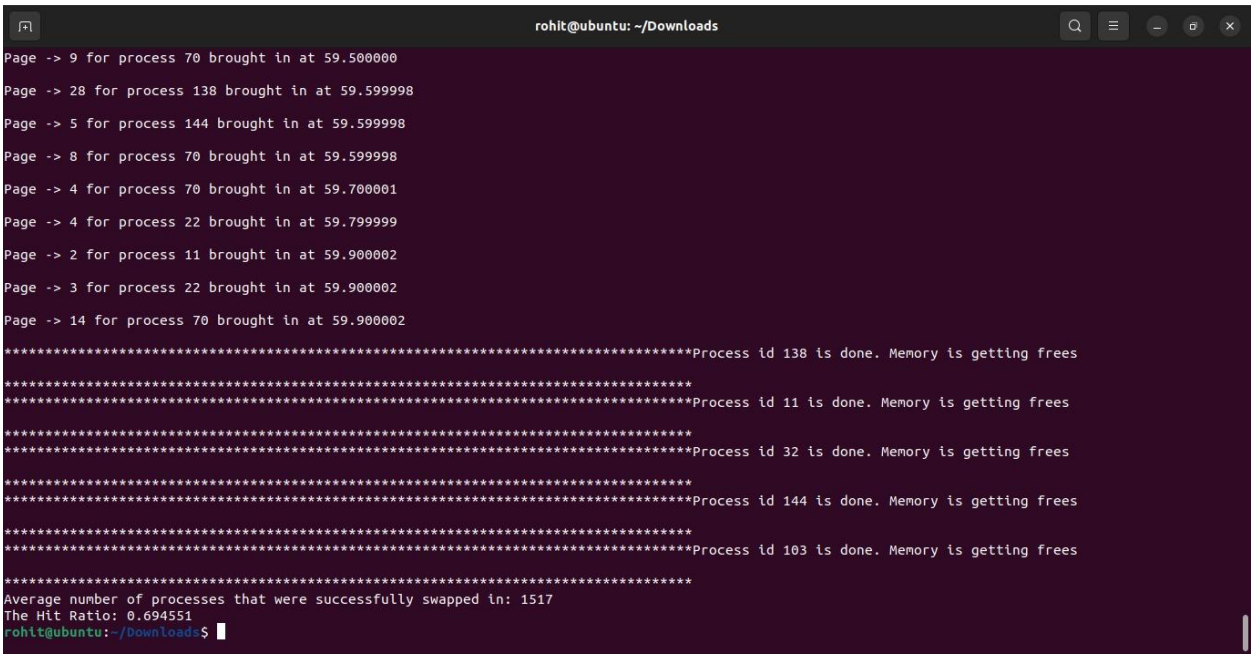
Least Recently Used



Most Frequently Used



Random


rohit@ubuntu: ~/Downloads
Page -> 9 for process 70 brought in at 59.500000
Page -> 28 for process 138 brought in at 59.599998
Page -> 5 for process 144 brought in at 59.599998
Page -> 8 for process 70 brought in at 59.599998
Page -> 4 for process 70 brought in at 59.700001
Page -> 4 for process 22 brought in at 59.799999
Page -> 2 for process 11 brought in at 59.900002
Page -> 3 for process 22 brought in at 59.900002
Page -> 14 for process 70 brought in at 59.900002
**********Process id 138 is done. Memory is getting frees
****
**********Process id 11 is done. Memory is getting frees
****
**********Process id 32 is done. Memory is getting frees
****
**********Process id 144 is done. Memory is getting frees
****
**********Process id 103 is done. Memory is getting frees
****
Average number of processes that were successfully swapped in: 1517
The Hit Ratio: 0.694551
rohit@ubuntu:~/Downloads$