COEN383
Advanced Operating System
Project 4 Report
Group No. 5
Title: Swapping and Paging Algorithms Simulation

Group Members:
1. Bharti Prakash - W1652174
2. Iniyan Chandran Ramachandran - W1651510
3. Ruthu Rajendra - W1653722
4. Siddhi Sanjay Powar - W1650216
5. Vaibhav Sachdeva - W1650084

Introduction
The project explores memory management algorithms for swapping and paging through a simulation in Java. The simulation generates random processes with specific characteristics, allocates memory, and applies page replacement strategies such as FIFO, LRU, LFU, MFU, and Random Pick.

Code Structure Overview
Page.java:
The Page class represents a page in the simulation, encapsulating information such as process ID, page number, arrival time, and references. It provides methods for managing pages in memory, checking free pages, updating page lists, and more.

Process.java:
The Process class defines the characteristics of a process, including process ID, size in pages, arrival time, service duration, and the current page it is referencing.

FirstComeFirstServe.java:
Implements the First Come First Serve page replacement algorithm, selecting the oldest page in memory for eviction.

LeastFrequentlyUsed.java:
Implements the Least Frequently Used page replacement algorithm, evicting the page with the least number of references.

LeastRecentlyUsed.java:
Implements the Least Recently Used page replacement algorithm, evicting the page that was least recently referenced.

MostFrequentlyUsed.java:
Implements the Most Frequently Used page replacement algorithm, evicting the page with the highest number of references.

RandomPick.java:
Implements the Random Pick page replacement algorithm, randomly selecting a page for eviction.

Main.java:
The main class orchestrates the simulation. It generates processes, initiates their execution, and evaluates the performance of different page replacement algorithms over multiple iterations.

Simulation Workflow

Workload Generation

1. **Process Generation**: Processes are generated with random sizes and durations.
2. **Job Queue**: Processes are added to a Job Queue linked list, sorted based on arrival time.
3. **Memory Allocation**: Memory is allocated using a linked list structure for free pages.
4. **Process Initialization**: Jobs at the head of the Job Queue are assigned memory if at least 4 free pages are available.

Paging

1. **Memory Structure**: Free pages are structured as a linked list.
2. **Memory Allocation**: Jobs are assigned memory if at least 4 free pages are available.
3. **Page-In from Disk**: When a process references a page not in memory, it is paged in from disk.

Process Execution

1. **Process Start**: A process always starts at page 0.
2. **Locality of Reference**: Processes reference memory locations using a locality of reference algorithm.
3. **Memory Reference Statistics**: For each memory reference, statistics are collected, including timestamp, process name, page referenced, page-in-memory status, and potential eviction details.
4. **Independent Execution**: Once in memory, processes run independently and simultaneously for their durations.

Simulation Execution

1. **Page Replacement Simulation**: The simulation runs different page replacement algorithms (FIFO, LRU, LFU, MFU, Random Pick) for one minute each, with results averaged over 5 runs.
2. **Detailed Records**: For each algorithm, detailed records are printed, including memory maps and information about process swaps.

Results and Analysis

The project evaluates the average hit/miss ratio and the number of successfully swapped-in processes for each page replacement algorithm over multiple iterations. The simulation provides insights into the efficiency and performance of these algorithms under different workloads.

Conclusion

The project successfully implements and evaluates memory management algorithms for swapping and paging. The simulation provides insights into the performance of different page replacement strategies under varying workloads.