LMS Assignment Report

Course: ICS1313 - Operating System Practices Laboratory
Assignment No: 9
Name: Simiyon Vinscent Samuel L
Reg No: 3122247001062

Assignment Title

Page Replacement Algorithms: Optimal and LRU Implementation

Aim / Learning Objectives

- To implement and analyze the Optimal and Least Recently Used (LRU) page replacement algorithms for virtual memory management
- To understand how page replacement algorithms minimize page faults in memory management
- To compare performance metrics between different page replacement strategies
- To demonstrate practical application of memory management concepts in operating systems

Theory

Page replacement algorithms are essential components of virtual memory systems that determine which memory page should be replaced when a new page needs to be loaded into memory. The primary objective is to minimize page faults and optimize system performance.

Optimal Algorithm: This algorithm replaces the page that will not be used for the longest period in the future. While it provides the theoretical minimum number of page faults, it requires future knowledge of the reference string, making it impractical for real implementation but useful as a benchmark.

LRU Algorithm: The Least Recently Used algorithm replaces the page that has been unused for the longest time. This algorithm is based on the principle of temporal locality, assuming that recently accessed pages are more likely to be accessed again soon.

Algorithm

Optimal Page Replacement:

1. Initialize page frames and reference string
2. For each page in reference string:
    - If page exists in memory, continue
    - If memory has free space, allocate page
    - Else, find page that will be used farthest in future
    - Replace that page with current page
3. Count total page faults

LRU Page Replacement:

1. Initialize page frames and reference string
2. For each page in reference string:
    - If page exists in memory, update its timestamp
    - If memory has free space, allocate page with current timestamp
    - Else, find page with oldest timestamp
    - Replace oldest page with current page
3. Count total page faults

Test Cases

Test Case	Reference String	Frame Size	Expected Page Faults (Optimal)	Expected Page Faults (LRU)
Test 1	7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1	3	9	12
Test 2	1,2,3,4,1,2,5,1,2,3,4,5	4	6	8
Test 3	2,3,2,1,5,2,4,5,3,2,5,2	3	6	7

Output Screenshots

[Screenshots would be inserted here showing program execution with both algorithms]

Observations

1. **Performance Comparison:** The Optimal algorithm consistently produces fewer page faults compared to LRU, as expected from theoretical analysis
2. **Practical Implementation:** LRU is implementable in real systems using timestamps or counters, while Optimal remains theoretical
3. **Memory Efficiency:** Both algorithms show improved performance with larger frame sizes
4. **Reference Pattern Impact:** Sequential access patterns favor both algorithms, while random patterns increase page fault rates

Learning Outcomes

- Successfully implemented both Optimal and LRU page replacement algorithms
- Analyzed performance differences between theoretical optimal and practical LRU approaches
- Understood the trade-offs between algorithm complexity and implementation feasibility
- Gained practical experience with memory management simulation and performance measurement
- Learned to evaluate algorithm efficiency using page fault metrics

Conclusion

This assignment provided hands-on experience with fundamental page replacement algorithms used in operating system memory management. The implementation demonstrated that while the Optimal algorithm provides superior performance, the LRU algorithm offers a practical compromise between performance and implementability in real systems.

References

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. Wiley.
2. Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems. Pearson.