SYSC 4001A

Operating Systems

Assignment 3

Group Submission

Version 1

Professor: Gabriel Wainer

Rigel Landmeter-Blondin, 101297902
Abdullah Khan, 101305235

Part 1

Part 2 Report

In shared memory, we store an array containing the rubric marks, an int that is the student number of the exam being marked, an array that is used to keep track of which questions has been marked, a boolean that indicates whether the current exam has finished being marked, and another boolean that indicates whether all the exams have been marked.

When running the program, a value is passed through the argv parameter to indicate the number of TAs marking the exams. This number is stored in the variable “numTA”. The parent process loops through a for loop numTA times, forking a child process each loop. It then waits for each child process to terminate before ending the program.

Each child process enters an infinite while loop. Inside the while loop they go through two for loop. In the first for loop, the process loops through each of the entries in the rubric marks array in shared memory, randomly deciding each loop whether to change the entry or not. If it does decide to change the entry, the process adjusts the entry in the array and updates the rubric.txt file. Semaphores are used to prevent other child processes from adjusting the rubric at the same time.

In the second for loop, the process loops through each of the entries in the marked array in shared memory. If an entry contains a 0, that indicated that the associated question has not been marked yet. As the process proceeds to mark the question, the associated entry in the array is updated to a 1, alerting other processes that this question is in the process of being marked/already marked. Semaphores are used again to prevent multiple processes from claiming the same question at the same time.

When each process finishes checking all 5 questions, they enter a while loop until the exam has finished being marked. After that, the boolean value in shared memory “finishedExam”, is updated to end the while loop, allowing the processes to restart the loop. The next exam is loaded into shared memory, and the entire process repeats until exam 9999. After exam 9999 is finished being marked, the boolean value in shared memory “finished” is set to true, allowing the processes to reach a break statement, freeing them from the while loop and allowing them to exit.

Once all the child processes have been terminated, the parent process terminated the shared memory and semaphores, closes the rubric.txt file, and terminates.

Part 3

Q1:

i) 3 Frames Allocated

a) FIFO (First-In-First-Out)

- We can model the page reference string and allocated frames as the following timeline. In this scheme, a lower priority refers to its nearness to being replaced. So, an allocated frame with a priority of 1 will be replaced on the next page fault, while an allocated frame with a priority of 3 will be replaced only after 3 following page faults.

1-3. The first 3 page lookups result in 3 page faults (since the page table is empty):
- Page Fault Count = 3
- Remaining Page Reference String:
- 417, 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
415	1
305	2
502	3

4. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
417	3
305	1
502	2

5. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
417	3
305	1
502	2

6. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
417	2
415	3
502	1

7. The page lookup 502 **does not** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
417	2
415	3
502	1

8. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
417	1
415	2
518	3

9. The page lookup 417 **does not** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
417	1
415	2
518	3

10. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
305	3
415	1
518	2

11. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
305	3
415	1
518	2

12. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 8
- Remaining Page Reference String:
- 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
305	2
502	3
518	1

13. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
305	1
502	2
520	3

14. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 10
- Remaining Page Reference String:
- 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	3
502	1
520	2

15. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 11
- Remaining Page Reference String:
- 305, 502, 415, 520, 518

Allocated Frames	Priority
518	2
417	3
520	1

16. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 12
- Remaining Page Reference String:
- 502, 415, 520, 518

Allocated Frames	Priority
518	1
417	2
305	3

17. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 13
- Remaining Page Reference String:
- 415, 520, 518

Allocated Frames	Priority
502	3
417	1
305	2

18. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 14
- Remaining Page Reference String:
- 520, 518

Allocated Frames	Priority
502	2
415	3
305	1

19. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 15
- Remaining Page Reference String:
- 518

Allocated Frames	Priority
502	1
415	2
520	3

20. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 16
- Remaining Page Reference String:

Allocated Frames	Priority
518	3
415	1
520	2

Hit Ratio:

$$Hit\ Ratio: \frac{\#\ Page\ Lookup\ Hits}{Total\ \#\ of\ Page\ Lookups}$$

$$Hit\ Ratio: \frac{4}{20}\ (20\%)$$

b) LRU (Least Recently Used)

- Similarly to the FIFO implementation, we can model the page reference string and allocated frames as a timeline, however, the priority is changed by using a counter which increments on every page lookup and resets if a page is used during a lookup. The allocated frame with the highest number is selected as the victim to be replaced.

1-3. The first 3 page lookups result in 3 page faults (since the page table is empty):
- Page Fault Count = 3
- Remaining Page Reference String:
- 417, 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	2
305	1
502	0

4. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
417	0
305	2
502	1

5. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
417	1
305	0
502	2

6. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
417	2
305	1
415	0

7. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
502	0
305	2
415	1

8. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
502	1
518	0
415	2

9. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
502	2
518	1
417	0

10. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	0
518	2
417	1

11. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 10
- Remaining Page Reference String:
- 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	1
415	0
417	2

12. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 11
- Remaining Page Reference String:
- 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	2
415	1
502	0

13. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 12
- Remaining Page Reference String:
- 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
520	0
415	2
502	1

14. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 13
- Remaining Page Reference String:
- 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
520	1
518	0
502	2

15. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 14
- Remaining Page Reference String:
- 305, 502, 415, 520, 518

Allocated Frames	Counter
520	2
518	1
417	0

16. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 15
- Remaining Page Reference String:
- 502, 415, 520, 518

Allocated Frames	Counter
305	0
518	2
417	1

17. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 16
- Remaining Page Reference String:
- 415, 520, 518

Allocated Frames	Counter
305	1
502	0
417	2

18. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 17
- Remaining Page Reference String:
- 520, 518

Allocated Frames	Counter
305	2
502	1
415	0

19. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 18
- Remaining Page Reference String:
- 518

Allocated Frames	Counter
520	0
502	2
415	1

20. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 19
- Remaining Page Reference String:

Allocated Frames	Counter
520	1
518	0
415	2

Hit Ratio:

$$Hit\ Ratio\colon \frac{\#\ Page\ Lookup\ Hits}{Total\ \#\ of\ Page\ Lookups}$$

$$Hit\ Ratio\colon \frac{1}{20}\ (5\%)$$

c) Optimal

- Similarly to the FIFO and LRU implementation, we can model the page reference string and allocated frames as a timeline, however, each allocated page keeps track of its distance to its next use in the page sequence. The one with the largest distance is used as the victim when a location is needed during a page fault.

1-3. The first 3 page lookups result in 3 page faults (since the page table is empty):
- Page Fault Count = 3
- Remaining Page Reference String:
- 417, 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	3
305	2
502	4

4. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	2
305	1
417	5

5. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	1
305	5
417	4

6. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	5
305	4
417	3

7. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
502	5
305	3
417	2

8. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	6
305	2
417	1

9. The page lookup 417 **does not** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	5
305	1
417	6

10. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	4
305	6
417	5

11. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	3
415	7
417	4

12. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 8
- Remaining Page Reference String:
- 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	2
502	5
417	3

13. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	1
520	6
417	2

14. The page lookup 518 **does not** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	6
520	5
417	1

15. The page lookup 417 **does not** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 305, 502, 415, 520, 518

Allocated Frames	Distance Away
518	5
520	4
417	Inf

16. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 10
- Remaining Page Reference String:
- 502, 415, 520, 518

Allocated Frames	Distance Away
518	4
520	3
305	Inf

17. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 11
- Remaining Page Reference String:
- 415, 520, 518

Allocated Frames	Distance Away
518	3
520	2
502	Inf

18. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 12
- Remaining Page Reference String:
- 520, 518

Allocated Frames	Distance Away
518	2
520	1
415	Inf

19. The page lookup 520 **does not** cause a page fault.
- Page Fault Count = 12
- Remaining Page Reference String:
- 518

Allocated Frames	Distance Away
518	1
520	Inf
415	Inf

20. The page lookup 518 **does not** cause a page fault.
- Page Fault Count = 12
- Remaining Page Reference String:

Allocated Frames	Distance Away
518	Inf
520	Inf
415	Inf

Hit Ratio:

$$Hit\ Ratio: \frac{\#\ Page\ Lookup\ Hits}{Total\ \#\ of\ Page\ Lookups}$$

$$Hit\ Ratio: \frac{8}{20}\ (40\%)$$

ii) 4 Frames Allocated

a) FIFO

1-4. The first 4 page lookups result in 4 page faults (since the page table is empty):
- Page Fault Count = 4
- Remaining Page Reference String:
- 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
415	1
305	2
502	3
417	4

5. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
415	1
305	2
502	3
417	4

6. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
415	1
305	2
502	3
417	4

7. The page lookup 502 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
415	1
305	2
502	3
417	4

8. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	4
305	1
502	2
417	3

9. The page lookup 417 **does not** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:

305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	4
305	1
502	2
417	3

10. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	4
305	1
502	2
417	3

11. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	3
415	4
502	1
417	2

12. The page lookup 502 **does not** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:
- 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	3
415	4
502	1
417	2

13. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	2
415	3
520	4
417	1

14. The page lookup 518 **does not** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 417, 305, 502, 415, 520, 518

Allocated Frames	Priority
518	2
415	3
520	4
417	1

15. The page lookup 417 **does not** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 305, 502, 415, 520, 518

Allocated Frames	Priority
518	2
415	3
520	4
417	1

16. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 8
- Remaining Page Reference String:
- 502, 415, 520, 518

Allocated Frames	Priority
518	1
415	2
520	3
305	4

17. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 415, 520, 518

Allocated Frames	Priority
502	4
415	1
520	2
305	3

18. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 520, 518

Allocated Frames	Priority
502	4
415	1
520	2
305	3

19. The page lookup 520 **does not** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 518

Allocated Frames	Priority
502	4
415	1
520	2
305	3

20. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 10
- Remaining Page Reference String:

Allocated Frames	Priority
502	3
518	4
520	1
305	2

Hit Ratio:

$$Hit\ Ratio: \frac{\#\ Page\ Lookup\ Hits}{Total\ \#\ of\ Page\ Lookups}$$

$$Hit\ Ratio: \frac{10}{20}\ (50\%)$$

b) LRU

1-4. The first 4 page lookups result in 4 page faults (since the page table is empty):
- Page Fault Count = 4
- Remaining Page Reference String:
- 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	3
305	2
502	1
417	0

5. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	4
305	0
502	2
417	1

6. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	0
305	1
502	3
417	2

7. The page lookup 502 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	1
305	2
502	0
417	3

8. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	2
305	3
502	1
518	0

9. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 6
- Remaining Page Reference String:

305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
415	3
417	0
502	2
518	1

10. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 7
- Remaining Page Reference String:
- 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	0
417	1
502	3
518	2

11. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 8
- Remaining Page Reference String:
- 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	1
417	2
415	0
518	3

12. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 9
- Remaining Page Reference String:
- 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	2
417	3
502	0
518	4

13. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 10
- Remaining Page Reference String:
- 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	3
417	4
502	1
520	0

14. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 11
- Remaining Page Reference String:
- 417, 305, 502, 415, 520, 518

Allocated Frames	Counter
305	4
518	0
502	2
520	1

15. The page lookup 417 **does** cause a page fault.
- Page Fault Count = 12
- Remaining Page Reference String:
- 305, 502, 415, 520, 518

Allocated Frames	Counter
417	0
518	1
502	3
520	2

16. The page lookup 305 **does** cause a page fault.
- Page Fault Count = 13
- Remaining Page Reference String:
- 502, 415, 520, 518

Allocated Frames	Counter
417	1
518	2
305	0
520	3

17. The page lookup 502 **does** cause a page fault.
- Page Fault Count = 14
- Remaining Page Reference String:
- 415, 520, 518

Allocated Frames	Counter
417	2
518	3
305	1
502	0

18. The page lookup 415 **does** cause a page fault.
- Page Fault Count = 15
- Remaining Page Reference String:
- 520, 518

Allocated Frames	Counter
417	3
415	0
305	2
502	1

19. The page lookup 520 **does** cause a page fault.
- Page Fault Count = 16
- Remaining Page Reference String:
- 518

Allocated Frames	Counter
520	0
415	1
305	3
502	2

20. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 17
- Remaining Page Reference String:

Allocated Frames	Counter
520	1
415	2
518	0
502	3

Hit Ratio:

$$Hit\ Ratio: \frac{\#\ Page\ Lookup\ Hits}{Total\ \#\ of\ Page\ Lookups}$$

$$Hit\ Ratio: \frac{3}{20}\ (15\%)$$

c) Optimal

1-4. The first 4 page lookups result in 4 page faults (since the page table is empty):
- Page Fault Count = 4
- Remaining Page Reference String:
- 305, 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	2
305	1
502	3
417	5

5. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 415, 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	1
305	5
502	2
417	4

6. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 502, 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	5
305	4
502	1
417	3

7. The page lookup 502 **does not** cause a page fault.
- Page Fault Count = 4
- Remaining Page Reference String:
- 518, 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	4
305	3
502	5
417	2

8. The page lookup 518 **does** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 417, 305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	3
305	2
518	6
417	1

9. The page lookup 417 **does not** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
305, 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	2
305	1
518	5
417	6

10. The page lookup 305 **does not** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 415, 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	1
305	6
518	4
417	5

11. The page lookup 415 **does not** cause a page fault.
- Page Fault Count = 5
- Remaining Page Reference String:
- 502, 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
415	7
305	5
518	3
417	4

12. The page lookup 502 **does** cause a page fault.
    - Page Fault Count = 6
    - Remaining Page Reference String:
    - 520, 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
502	5
305	4
518	2
417	3

13. The page lookup 520 **does** cause a page fault.
    - Page Fault Count = 7
    - Remaining Page Reference String:
    - 518, 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
520	6
305	3
518	1
417	2

14. The page lookup 518 **does not** cause a page fault.
    - Page Fault Count = 7
    - Remaining Page Reference String:
    - 417, 305, 502, 415, 520, 518

Allocated Frames	Distance Away
520	5
305	2
518	6
417	1

15. The page lookup 417 **does not** cause a page fault.
    - Page Fault Count = 7
    - Remaining Page Reference String:
    - 305, 502, 415, 520, 518

Allocated Frames	Distance Away
520	4
305	1
518	5
417	Inf

16. The page lookup 305 **does not** cause a page fault.
    - Page Fault Count = 7
    - Remaining Page Reference String:
    - 502, 415, 520, 518

Allocated Frames	Distance Away
520	3
305	Inf
518	4
417	Inf

17. The page lookup 502 **does** cause a page fault.
    - Page Fault Count = 8
    - Remaining Page Reference String:
    - 415, 520, 518

Allocated Frames	Distance Away
520	2
502	Inf
518	3
417	Inf

18. The page lookup 415 **does** cause a page fault.
    - Page Fault Count = 9
    - Remaining Page Reference String:
    - 520, 518

Allocated Frames	Distance Away
520	1
415	Inf
518	2
417	Inf

19. The page lookup 520 **does not** cause a page fault.
    - Page Fault Count = 9
    - Remaining Page Reference String:
    - 518

Allocated Frames	Distance Away
520	Inf
415	Inf
518	1
417	Inf

20. The page lookup 518 **does not** cause a page fault.
    - Page Fault Count = 9
    - Remaining Page Reference String:

Allocated Frames	Distance Away
520	Inf
415	Inf
518	Inf
417	Inf

Hit Ratio:

$$Hit\ Ratio: \frac{\#\ Page\ Lookup\ Hits}{Total\ \#\ of\ Page\ Lookups}$$

$$Hit\ Ratio: \frac{11}{20}\ (55\%)$$

iii) Conclusion, and questions

- From this one experiment, the FIFO algorithm performed better with a 20% hit rate compared to LRU's 5% hit rate. However, this disparity is mostly set by this example's sequence and not truly indicative of the real-world comparison. At a page table size of 3, the performance is heavily reliant on the peculiar sequence since it doesn't allow the algorithms to truly show their differences. When moving to a 4-frame sized table, the FIFO algorithm still beats out LRU at a 50% hit rate compared to a 15% hit rate. The LRU should normally outperform FIFO in real-world sequences, where it is common to access pages near each other giving LRU the edge over FIFO. In some cases, FIFO can exhibit Belady's Anomaly where LRU will never. If we include the optimal algorithm in the comparison, it will always match or beat other algorithms since it guarantees the minimal amount of page faults. However, in real systems, it is impractical to implement since it requires knowledge of future events. This is only possible in very specific situations where everything is preplanned. Comparing the LRU and FIFO algorithms, FIFO can sometimes outperform LRU where a page sequence contains a pseudorandom cyclical behavior a size slightly bigger than the page table. The bias created by LRU can cause the incorrect page to be removed where the order of FIFO reduces the fault rate. However, in sequences where common pages are used (locality) frequently, LRU will outperform FIFO by a wide margin.

Q2

a)

- Since the page table is itself stored in physical memory, an initial lookup in main memory is required to find the corresponding physical address of the logical address. Then, a second lookup with the newly found logical address is completed. So, the lookup time without a TLB would be 240ns (2 * 120ns).

b)

- Assuming the TLB search happens sequentially to the physical memory lookup (the page lookup in the TLB and page table are not done at the same time), the Effective Access Time (EAT) can be calculated using the following equation, where t is the lookup time for the physical memory, ε is the lookup time for the TLB, and α is the hit ratio for the TLB:

$$EAT = (1 - \alpha) * (t + \varepsilon) +* \alpha * \varepsilon + t$$

$$EAT = 2t + \varepsilon - \alpha * t$$

(Slightly different initial equation compared to the slides but identical final equation. The $(1 - \alpha) * (t + \varepsilon)$ term is the case where there is a TLB miss. Inversely, the $\alpha * \varepsilon$ term is when there is a TLB hit. The additional t is the required lookup time when the physical address is acquired.)

Using that equation, the EAT can be calculated using the following values: $\alpha = .95$, $\varepsilon = 20$, and $t = 120$.

$$EAT = 2t + \varepsilon - \alpha * t$$

$$EAT = 2(120) + 20 - 0.95 * 120$$

$$EAT = 146ns$$

In conclusion, with the addition of the TLB, the effective memory access time would be 146ns.

c)

- Generally, adding a TLB improves performance by drastically reducing the lookup time on pages that are frequently searched for. In real systems there could be specific pages that are required often, and reducing those lookup times can dramatically improve the effective access time. However, if the page look ups don't require fetching reoccurring pages, the TLB becomes pure overhead. If we take the example from part b of this question, having a hit rate lower than 50% would result in the EAT being worse than not having the TLB.

Q3:

Consider a system with a paged logical address space composed of 128 pages of 4 Kbytes each, mapped into a 512 Kbytes physical memory space. Answer the following questions and justify your answers.

(a) What is the format and size (in bits) of the processor's logical address?

Page Number:

$$\log_2(128) = 7$$

Page Number is 7 bits

Page Offset:

4 Kbytes = 4096 bytes

$$\log_2(4096) = 12$$

Page Offset is 12 bits

The logical address has a size of 19 bits. 7 bits for the page number and 12 bits for the page offset.

(b) What is the required length (number of entries) and width (size of each entry in bits, disregarding control bits) of the page table?

Number of entries in the page table is the same as number of pages: 128

Width of each entry in page table:

Number of frames:

$$\frac{512\ kB}{4\ kB} = 128$$

(Space on physical memory divided by page size gives 128 frames)

$$\log_2(128) = 7$$

Each entry has a width of 7 bits.

(c) What is the effect on the page table width if now the physical memory space is reduced by half (from 512 Kbytes to 256 Kbytes)? Assume that the number of page entries and page size remain the same.

By halving the amount of physical memory space, the number of frames also gets halved.

$$\frac{256\ kB}{4\ kB} = 64$$

Since the number of frames has been halved, the number of bits in the page table width reduces by 1.

$$\log_2 \quad 64 = 6$$

Q4:

Explain, in detail, the sequence of operations and file system data structure accesses that occur when a process executes the lseek(fd, offset, SEEK_END) system call. Consider a system using a hierarchical directory structure and assume the file described by the file descriptor (fd) is not currently open by any other process.

When lseek is called, the open file table is accessed to check for fd. If fd is located within the open file table, the system accesses the file's inode and the current position pointer from the fd entry in the table. From the inode, it retrieves the size of the file. The offset value is then added to the size value, creating a new offset value for the current position pointer. The current position pointer is then set to this new offset. Finally, the lseek syscall returns the new position of the pointer to the process.

Q5:

(a) Consider a file system that uses inodes to represent files. Disk blocks are 8Kb in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

Size of direct disk blocks:

$$12 \cdot 8\, kB = 96\, kB$$

Number of pointers in an indirect block:

$$\frac{8192\ bytes}{4\ bytes} = 2048$$

Single indirect disk blocks size:

$$2048^1 = 2048$$

$$2048 \cdot 8\, Kb = 16384\, kB$$

Double indirect disk blocks size:

$$2048^2 = 4194304$$

$$4194304 \cdot 8\, Kb = 33554432\, kB$$

Triple indirect disk blocks size:

$$2048^3 = 8589934592$$

$$8589934592 \cdot 8\, kB = 68719476736\, kB$$

96 kB + 16384 kB + 33554432 kB + 68719476736 kB = 68753047648 kB

The maximum size of a file that can be stored is 68753047648 kB or about 70.4 TB.

(b) Explain what you can do in case (a) if you need to store a file that is larger than the maximum size computed. Give an example showing how you can define a larger file, and what the size of that file would be.

If we need to store a file that is larger than the maximum size computed (about 70.4 TB), we could define an inode that also contains a quadruple indirect disk block.

Quadruple indirect disk blocks size:

$$2048^4 = 1.759218604e13$$
$$1.759218604e13 \cdot 8\ kB = 1.407374884e14\ kB$$

Adding a quadruple indirect disk block would increase the file size by about 1.407374884e14 kB or 144115 TB, increasing the maximum size of the file to about 144185 TB or about 144 PB.

SYSC 4001 – Assignment Part 1 – Report

Group Members:

Rigel Landmeter-Blondin, 10197902

Abdullah Khan, 101305235

Repository Links:

Part 1:

https://github.com/RigelLB/SYSC4001_A3_P1

Part 2:

https://github.com/AbdullahKhan28CU/SYSC4001_A3P2

Simulator Results:

To preface this report, midway through doing the test cases and analyzing them, I have noticed that my choice of test for the respective cases of I/O bound, CPU bound, and similar bursts are not ideal for a real test, however, I ran out of time to modify the whole report and test to fix this issue, so, the rest of this report is based on the tests previously made. I also realize that the python script I used to gather the metrics my be slightly faulty as some of the values collected look “off”.

The metrics for the throughput, average wait time, average turnaround time, and average response time were collected as follows.

Throughput: $\frac{\#\ processes}{Total\ Time}$

Average Wait Time: Sum of time spent in the READY state

Average Turnaround Time: The time spent between the NEW state to the TERMINATED state

Average Response Time: The time spent between the NEW state to the first RUNNING state.

All the values were then summed for all the test cases in the same category is divided by the number of respective test cases. The different algorithms result in the following data table:

EP

Test #	I/O Bound	CPU Bound	Similar I/O and CPU Burst	Throughput	Average Wait Time	Average Turnaround Time	Average Response Time
			16	0.007	277	447	199
			6	0.007	288	458	233
				0.007	282.5	452.5	216
		11		0.008	125	245.67	125
		3		0.007	50	200	50
		4		0.007	45	200	2.5
		8		0.004	100	350	100
				0.0065	80	248.9175	69.375
	10			0.007	32.5	240	2.5
	5			0.007	110	278.33	15
				0.007	71.25	259.165	8.75

RR

Test #	I/O Bound	CPU Bound	Similar I/O and CPU Burst	Throughput	Average Wait Time	Average Turnaround Time	Average Response Time
			16	0.007	261	431	75
			6	0.007	261	431	75
				0.007	261	431	75
		11		0.008	135.33	240	2.5
		3		0.007	50	200	50
		4		0.007	45	200	2.5
		8		0.004	200	450	38.33
				0.0065	107.5825	272.5	23.3325
	10			0.007	32.5	240	2.5
	5			0.007	115	283.33	21.67
				0.007	73.75	261.665	12.085

EP_RR

Test #	I/O Bound	CPU Bound	Similar I/O and CPU Burst	Throughput	Average Wait Time	Average Turnaround Time	Average Response Time
			16	0.007	265	435	187
			6	0.007	306	476	227
				0.007	285.5	455.5	207
		11		0.008	135.33	256	71.67
		3		0.007	50	200	50
		4		0.007	82.5	237.5	2.5
		8		0.004	200	450	0
				0.0065	116.9575	285.875	31.0425
	10			0.006	77.5	285	2.5
	5			0.008	143.33	237.5	1.67
				0.007	110.415	261.25	2.085

The different colors are used to separate different groups of tests from each other. For example, the first set of green values are the collective averages of the Similar I/O and CPU Burst test group using the External Priorities algorithm.

From the collected data, the waiting time for similar IO/CPU bursts are similar, with the slight edge to the round robin algorithm. In terms of response times, the round-robin algorithm beats out the two other algorithms by a wide margin. In theory, the response time for the external priority algorithm should be worse than the two other algorithms since processes run until completion before handing over the CPU, which we do see in the simulation results. The hybrid external priority w/ the round-robin timeout should have slightly better response times since it gives up the CPU with higher priority processes when they arrive and are forced out by the quantum timer, which we do see in the data. In general, RR will be very good at reducing the waiting time and response time in I/O bound situations since it gives the opportunity to all processes to use the CPU equally. However, in this simulation it is not observed. This could be due to the test cases and the relatively large quantum. Since most of the processes don't have CPU processing times that are very large, they will very rarely be handled by the round robin style timeout. In practice, however, the hybrid approach can be very powerful and outshine the other algorithm in most cases. The ability to attribute priorities can be very important since some processes will be more important than others. The downsides such as the response time can be attributed somewhat by modifying the quantum to obtain the best performance.