Theory Assignment 3
COEN 346

Adrian PATTERSON
ID: 40048841

December 7, 2020

Date Due:	December 6, 2020
Instructor:	Professor Goodarzi
Section:	F

I certify that this submission is my original work and meets the Faculty's Expectations of Originality.

Université Concordia University
GINA CODY SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

1. Question 1

   (a) **What is (are) the advantage(s) and/or disadvantage(s) of small versus big page sizes?**

      i. Advantages of Small Page Sizes over Big Page Sizes
      A system with small page sizes implies that this system will use a large page table. Consequently, a small page size system will decrease the chances of internal fragmentation compared to a large page size, as large page sizes can waste unnecessary space.

      ii. Disadvantages of Small Page Sizes over Big Page Sizes
      The overall amount of page faults will be greater for a system with small page sizes, as compared with a big page size system. The amount of page faults are greater due to the large page table size associated with a small page size. With large page tables comes a higher amount of page faults. Finally, smaller page sizes also increase the amount of overhead associated with reading and writing pages.

   (b) **What is (are) the advantage(s) of paging over segmentation?**
   The main advantage that paging holds over segmentation is that paging is faster than segmentation. Since pages and frames are of equal size, disk access is easier and faster. Another advantage is that paging offers no external fragmentation, while segmentation does.

   (c) **What is (are) the advantage(s) of segmentation over paging?**
   The main advantage of using segmentation over paging is memory. Segmentation needs less memory than paging. Another advantage of segmentation over paging is that segmentation offers no internal fragmentation, while paging does.

2. Question 2

   (a) **What are the critical sections inside the wait and signal operations which are protected by disabling and enabling of interrupts?**
   The critical sections within the wait and signal operations that are protected by the enabling and disabling of interrupts are the increment and decrement operations on `sem.value`, as well as the if statements that follow. Interrupts are disabled to allow `sem.value` to be incremented and decremented atomically.

   (b) **Give example of a specific execution scenario for the above code leading to inconsistency if the critical sections inside implementation of wait() and signal() are not protected (by disabling of interrupts).**
   A scenario where critical sections are inconsistently affected due to the lack of atomicity (without disabling interrupts) would be if a process that is en-queued is immediately de-queued. For example, suppose the wait function starts execution and the `sem.value` is en-queued. At this point, an interrupt occurs where another process enters the body of the signal function. Here, the same enqueued process will be de-queued, resulting in the loss of this process. This is thus an inconsistency that can result if these operations are not protected by the enabling and disabling of interrupts.

   (c) **Suppose that process A calling semaphore wait() gets blocked and another process B is selected to run (refer to the above code). Since interrupts are enabled only at the completion of the wait operation, will B start executing with the interrupts disabled? Explain your answer.**
   B will not start executing, as process A has blocked process B within the wait function. B may attempt to execute, but since interrupts are disabled and B is blocked by A, nothing will happen. Only once A completes execution and re-enables interrupts, can process B begin its execution. In this scenario, all processes are in deadlock.

3. Question 3
Calculate the effective memory access time for the system.
Given:
TLB Access Time = 0.2 μs
Main Memory Access Time = 1μs
Page requests: 2% are faults, 98% are hits
Page Fault Time = 20ms
Of 98% page request hits, 80% of accesses are found in TLB, 20% are TLB misses

We know that Effective Memory Access Time
= TLB Hit Ratio(TLB Access Time + Main Memory Access Time)
+TLB Miss Ratio(TLB Access Time + 2(Main Memory Access Time))

With TLB Hit Ratio given by

$$\text{TLB Hit Ratio} = \frac{\text{TLB Accesses}}{\text{TLB Hits}} = \frac{80}{98} = 0.8163$$

Thus, Effective Access Time (EAT) is given by:

$$\text{EAT} = 0.8163(0.2\mu s + 1\mu s) + (1 - 0.8163)(0.2\mu s + 2(1\mu s)) = \mathbf{1.3837}\mu\mathbf{s}$$

Thus, the effective access time is 1.3837 microseconds.

4. Question 4

(a) **Show the memory representation of the pages using the LRU algorithm and an allocation of 3 frames. How many page faults are there?**

Least Recently Used

Reference String	0	3	2	0	1	1	0	1	3	3	6	7	6	7	0	1	3	2	4
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
F1	0	3	2	0	1	1	0	1	3	3	6	7	6	7	0	1	3	2	4
F2		0	3	2	0	0	1	0	1	1	3	6	7	6	7	0	1	3	2
F3			0	3	2	2	2	2	0	0	1	3	3	3	6	7	0	1	3
Hit	☐	☐	☐	☑	☐	☑	☑	☑	☐	☑	☐	☐	☑	☑	☐	☐	☐	☐	☐

Total Page Faults: 12

Figure 1: LRU Memory Representation

(b) **Show the memory representation of the pages using the Belady Optimal algorithm and an allocation of 3 frames. How many page faults are there?**

Belady Optimal

Reference String	0	3	2	0	1	1	0	1	3	3	6	7	6	7	0	1	3	2	4
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
F1	0	3	2	2	1	1	1	1	1	1	6	7	7	7	7	1	3	2	4
F2		0	3	3	3	3	3	3	3	3	1	6	6	6	6	6	6	6	6
F3			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hit	☐	☐	☐	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☐	☐	☐	☐

Total Page Faults: 10

Figure 2: Belady Optimal Memory Representation

(c) **Show the memory representation of the pages using the working set model with a window size Δ=3 (Δ indicates the maximum number allowed for a page to be in memory before being replaced; i.e. if a page is not used for 3 consecutive times, then it must either be used/demanded next, or it has to be removed). How many page faults are there?**

For this problem, I am assuming that this working-set algorithm uses prepaging, where pages can be loaded before they are needed. I.e., if a page is replaced, the next upcoming page will be put in place.

Working Set Model

Reference String		0	3	2	0	1	1	0	1	3	3	6	7	6	7	0	1	3	2	4
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
F1	0	3	2	0	1	1	0	1	3	3	6	7	6	7	0	1	3	2	4	
F2		0	3	2	0	0	1	0	1	1	3	6	7	6	7	0	1	3	2	
F3			0	3	2	3	3	3	0	0	7	3	3	0	6	7	0	1	3	
Hit	☐	☐	☐	☑	☐	☑	☑	☑	☑	☑	☐	☑	☑	☑	☑	☐	☐	☐	☐	
Removed:						2					1			3						
Total Page Faults:																			9	

Figure 3: Working Set Memory Representation

5. Question 5

   (a) **Advantage of a page table implementation**
   An advantage of using a page table implementation on a CPU is that swapping between page frames is relatively easy with equal sized pages and frames, which makes the operations cheaper and more efficient for a CPU.

   (b) **Disadvantage of a page table implementation**
   A disadvantage of this implementation is that the memory management algorithm uses a lookup table, which in turn increases memory access time.

6. Question 6

   (a) **Advantage of a global page replacement algorithm has over a local page replacement**
   A advantage a global page replacement has over local page replacement is the increased amount of page replacement options. In a global page replacement algorithm, memory can be allocated to pages which are outside of a process's allocated memory. On the other hand, in a local page replacement algorithm, a process is confined to solely the memory allocated to that process.

   (b) **Disadvantage of a global page replacement algorithm has over a local page replacement**
   A disadvantage a global page replacement over local page replacement is the complexity introduced by a global memory allocation scheme. In the case of a global replacement algorithm, processes must always refer to the globally allocated memory, possibly increasing time for memory allocation.

7. Question 7

   (a) **Largest file size (in bytes) that this disk allocation follows**
   Given:
   Disk block pointers are **2 bytes long**
   Data block size is 8 KB = $8 \times 2^{10}B = 2^{13}B$
   Each inode has **8 pointers to data blocks**
   1,2, and 3 level indexing pointers

From the given information, the following can be deduced:

With 8 pointers to data blocks, each inode has $8 \times 8KB = 2^3 \times 2^{13}B = 2^{16}B = 64KB$
Addresses per block of data = $\frac{2^{13}}{2} = 2^{12}B = 4 \times 2^{10}B = \mathbf{4KB}$
One level memory space = $8KB \times 4KB = 2^{13} \times 2^{12}B = 2^{25} = 32MB$
Two level memory space = $8KB \times 4KB^2 = 2^{13} \times 2^{24}B = 2^{37} = 128GB$
Three level memory space = $8KB \times 4KB^3 = 2^{13} \times 2^{36}B = 2^{49}B = 512TB$

Thus, the largest file size this disk allocation follows is

$$\text{Largest File Size} = 64KB + 32MB + 128GB + 512TB \approx \mathbf{512TB}$$

(b) **Is there enough space on the disk to hold the largest file? Explain.**
Given:
Disk block pointers are 2 bytes long= **16 bits**

From the given information, the following can be deduced:

Total disk size = Addressable blocks $\times$ Block Size

$= 2^{16} \times 8KB = 2^{16} \times 2^{13}B = 2^{19B} = 512MB$

Thus, the disk size is clearly not big enough to hold the largest possible file in this system.

8. Question 8
Some systems automatically open a file when it is referenced for the first time and close the file when the job terminates. Discuss the advantages and disadvantages of this scheme as compared to the more traditional one, where the user has to open and close the file explicitly.

(a) Advantages
The advantage of this systems is that the file IO is convenient for the user. The user can save time with the automatic opening and closing of files, as they no longer have to explicitly open and close them.

(b) Disadvantages
The disadvantages of this system come with the possible errors that could occur. For example, if a file were being written to while this system automatically opens it, it could result in the corruption of the file. Additionally, more overhead memory is required for automatic file opening, as files are opened even if they aren't needed.

9. Question 9

(a) **What is the difference between preemptive and non-preemptive scheduling? Why is strict non-preemptive scheduling unlikely to be used in a computer system that provides both batch and timesharing service?**
Preemptive scheduling allocates CPU to processes for different amounts of time, while non-preemptive scheduling allocates CPU to processes until termination. In other words, preemptive scheduled processes can be "interrupted," while non-preemptive scheduled processes run from start to finish. A batch and timesharing service is unlikely to use non-preemptive as a service of this kind requires a high level of multiprogrammation, which is limited by non-preemptive processes. Only one process can start and execute at a time in a non-preemptive system.

(b) **What is the trade-off used to select the quantum size, say, in pure Round-Robin scheduling?**
The trade off with quantum value is the following: For a larger quantum, processes are scheduled in a first come first serve manner; for a smaller quantum, processes are scheduled in a

shortest job first manner. Depending on the application, one of these trade-offs may be more desirable than the other.

10. Question 10

    What advantage is there in having different values of the scheduling quantum on different levels of a multilevel feedback queuing system?

    The advantage of having different quantum values over different levels of multilevel feedback queuing system is that the degree of multiprogramming is increased. Different quantums at different levels allow the system to be optimized for burst time, allowing the first level to run short quantum values and the lower levels to run longer quantums. This allows the turnaround time and response time to be optimized, allowing for a higher degree of multiprogramming.

11. Question 11

    (a) **Draw Gantt charts for the execution scenarios assuming:**

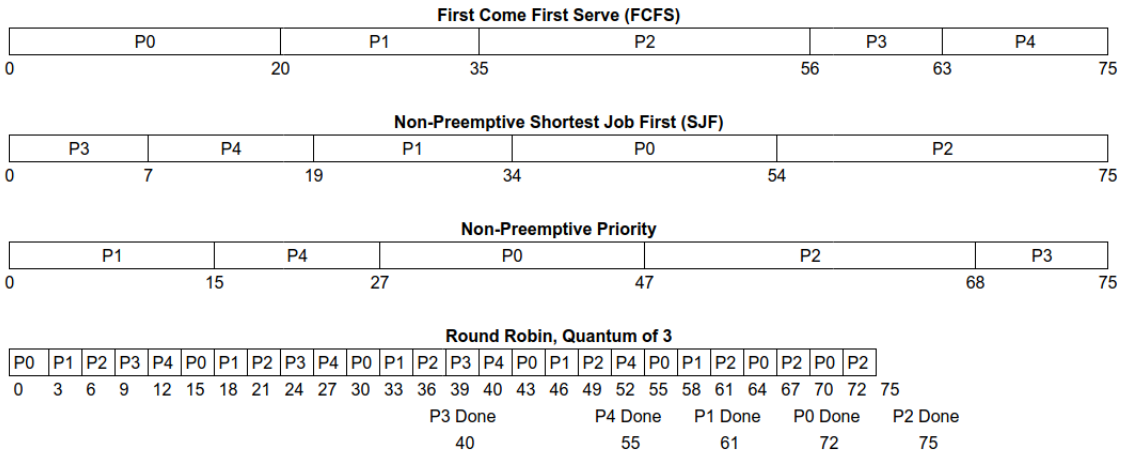

Figure 4: Scheduling Algorithm Gantt Charts

    (b) **What is the waiting time of each process in each case?**

        i. FCFS Waiting Times
            A. P0 Waiting Time: **0**
            B. P1 Waiting Time: **20**
            C. P2 Waiting Time: **35**
            D. P3 Waiting Time: **56**
            E. P4 Waiting Time: **63**
        ii. Non-Preemptive SJF Waiting Times
            A. P0 Waiting Time: **34**
            B. P1 Waiting Time: **19**
            C. P2 Waiting Time: **54**
            D. P3 Waiting Time: **0**
            E. P4 Waiting Time: **7**
        iii. Non-Preemptive Priority Waiting Times
            A. P0 Waiting Time: **27**
            B. P1 Waiting Time: **0**
            C. P2 Waiting Time: **47**
            D. P3 Waiting Time: **68**

E. P4 Waiting Time: **15**

iv. Round Robin Waiting Times

A. P0 Waiting Time: **52**

B. P1 Waiting Time: **46**

C. P2 Waiting Time: **54**

D. P3 Waiting Time: **33**

E. P4 Waiting Time: **43**

(c) **What is the response time of each process in each case?**

i. FCFS Response Times

A. P0 Response Time: **0**

B. P1 Response Time: **20**

C. P2 Response Time: **35**

D. P3 Response Time: **56**

E. P4 Response Time: **63**

ii. Non-Preemptive SJF Response Times

A. P0 Response Time: **34**

B. P1 Response Time: **19**

C. P2 Response Time: **54**

D. P3 Response Time: **0**

E. P4 Response Time: **7**

iii. Non-Preemptive Priority Response Times

A. P0 Response Time: **27**

B. P1 Response Time: **0**

C. P2 Response Time: **47**

D. P3 Response Time: **68**

E. P4 Response Time: **15**

iv. Round Robin Response Times

A. P0 Response Time: **0**

B. P1 Response Time: **3**

C. P2 Response Time: **6**

D. P3 Response Time: **9**

E. P4 Response Time: **12**

(d) **What is the turn-around time of each process in each case?**

i. FCFS Turn-Around Times

A. P0 Turn-Around Time: **20**

B. P1 Turn-Around Time: **35**

C. P2 Turn-Around Time: **56**

D. P3 Turn-Around Time: **63**

E. P4 Turn-Around Time: **75**

ii. Non-Preemptive SJF Turn-Around Times

A. P0 Turn-Around Time: **54**

B. P1 Turn-Around Time: **34**

C. P2 Turn-Around Time: **75**

D. P3 Turn-Around Time: **7**

E. P4 Turn-Around Time: **19**

iii. Non-Preemptive Priority Turn-Around Times

A. P0 Turn-Around Time: **47**

B. P1 Turn-Around Time: **15**
C. P2 Turn-Around Time: **68**
D. P3 Turn-Around Time: **75**
E. P4 Turn-Around Time: **27**

iv. Round Robin Turn-Around Times

A. P0 Turn-Around Time: **72**
B. P1 Turn-Around Time: **61**
C. P2 Turn-Around Time: **75**
D. P3 Turn-Around Time: **40**
E. P4 Turn-Around Time: **55**