Operating Systems - Exercise 4 - Answers to Theoretical Part

Yevgeni Dysin, Lilach Kelman

June 15, 2017

1 Part 1

1.

(a) Round Robin:

Time	Process
0-2	P1
2-4	P2
4-6	P1
6-8	P3
8-10	P4
10-11	P2
11-13	P1
13-14	P5
14-16	P4
16-18	P1
18-20	P4
20-22	P1
22-23	P4

Turnaround time: $\frac{(22+11CS)+(9+4CS)+(4+1CS)+(19+10CS)+(7+4CS)}{5} = 12.2 + 6CS$

Wait time: $\frac{(12+11CS)+(6+4CS)+(2+1CS)+(12+10CS)+(6+4CS)}{5} = 7.6 + 6CS$

(b) FCFS:

Time	Process
0-10	P1
10-13	P2
13-15	P3
15-22	P4
22-23	P5

Turnaround time: $\frac{10+(11+1CS)+(11+2CS)+(18+3CS)+(16+4CS)}{5} = 13.2 + 2CS$

Wait time: $\frac{0+(8+1CS)+(9+2CS)+(11+3CS)+(15+4CS)}{5} = 8.6 + 2CS$

(c) SRTF:

Time	Process
0-2	P1
2-5	P2
5-7	P3
7-8	P5
8-15	P4
15-23	P1

Turnaround time: $\frac{(23+5CS)+3+(3+CS)+(11+3CS)+1}{5} = 8.2 + 1.8CS$

Wait time: $\frac{(13+5CS)+0+(1+1CS)+(4+3CS)+0}{5} = 3.6 + 1.8CS$

(d) PS:

Time	Process
0-10	P1
10-13	P2
13-20	P4
20-22	P3
22-23	P5

Turnaround time: $\frac{10+(11+1CS)+(18+3CS)+(16+2CS)+(16+4CS)}{5} = 14.2 + 2CS$

Wait time: $\frac{0+(8+1CS)+(16+3CS)+(9+2CS)+(15+4CS)}{5} = 9.6 + 2CS$

(e) PS w\P:

Time	Process
0-2	P1
2-5	P2
5-12	P4
12-14	P3
14-22	P1
22-23	P5

Turnaround time: $\frac{(22+4CS)+(3+1CS)+(10+2CS)+(8+1CS)+(16+3CS)}{5} = 11.8 + 2.2CS$

Wait time: $\frac{(12+4CS)+(0+1CS)+(8+2CS)+(1+1CS)+(15+3CS)}{5} = 7.2 + 2.2CS$

2. Yes, mainly because in our implementation the cached blocks are saved on the heap - which means it's stored inside the main memory. As we saw numerous times during this course - main has a much faster access time than the physical disk. However, it is worth mentioning that there might occur a situation where we run out of main and the kernel might place some portions of our cache in virtual memory which is stored on the physical disk - thus slowing the access time.

3. The problem with implementing complex algorithms such as FBR is the amount of overhead that has to be performed upon each memory access. This is not a problem for when we R\W from the disk because the R\W operations are relatively slow but, it is a problem when we access the memory and try to retrieve\store cached data because memory access is a much faster operation.

4. Let's consider the case where we have 3 different objects and a cache capacity of 2.

   (a) LFU better than LRU in the following case: we have the following stream: [A, A, B, C, A]. In the case of LFU B will be evicted and we won't need to access the disk again after C because A already cached.
   With that stream LRU will evict A so it can cache C which means that there will be another disk access for the last A.

   (b) LRU better than LFU: Lets consider the following stream: [A,A,B,C,B]. If we use the LFU algorithm B will be evicted before reading C which means there will be another disk access before reading the last B.
   However, if we use LRU, we will evict A before reading C so, the when we read the last B we will read it from the cache.

   (c) Now, let's check the case where both algorithms fail: the received stream is [A,B,C,A].
   Here, both LFU and LRU will have to perform 2 evictions and 2 disk accesses for the last 2 objects in the stream.

5. If we increased the counter with each access to items in the new section, when we get to a place where an item isn't used for a long period of time and moved to the old section, it would be inserted with a access frequency which is higher than other items in the old section. This will mean we will evict items that are going to be accessed in the future but the item with the high count won't be evicted, even though it's not going to be used in the near future.

2 Part 2

2.1 Question 1

From the data given us by the question we can deduce that we will need a single indirect pointer to access the 40K-th byte. This operation will perform the following 9 disk access operations:

1. Access inode "/"
2. Read directory "/"
3. Access inode "/os/"
4. Read directory "/os/"
5. Access inode "/os/readme.txt"
6. Acces the single indirect pointer of our desire
7. Read the relevant block using the indirect pointer - the block that contains byte 40,000
8. Modify the block as we wish and write it to the disk
9. Update inode metadata

2.2 Question 2

There will be performed 2 software interrupts of type "trap" for the read and seek operations, bacause both of these functions perform Sys-Calls for their operation. Finally, there will be single hardware interrupt by the disk to inform the OS of finished read operation.

2.3 Question 3

1. Lets consider the following solution: 1 queue that will be managed by the RR algorithm with a quantum of 1 and and another queue that will be managed by the SRTF algorithm. Now, we define the following properties for our solution:

   (a) If job from queue 1 hasn't finished in 1 turn it will be moved to queue 2.

   (b) Queue 2 will have a lower priority than queue 1 and it can be preempted by queue 1 scheduling decisions.

   The proposed solution will ensure that jobs with short runtime will be able to finish their task before jobs with longer runtime. So, on average we will get a shorter wait time, as requested.

2. This algorithm has the potential of having high amount of preemptions thus, causing the process to have a high amount of context-switches. An alternative to our solution would be to use the FCFS algorithm which as we've shown before has a low amount of context-switches thus, having minimum overhead.