CS342 Operating Systems - Spring 2023
Exercise Homework #2 Solutions
(exercise version of homework 2 - not to be graded) Solutions

Assigned: May 28, 2023.
Due date: NA. Document version: 1.1

Q1. Consider a system consisting of 4 processes and a single resource type. The current state of the Maximum Demand (Claim) and Allocation matrices are as follows:

Max Demand	Allocation
P1 3	P1 1
P2 2	P2 1
P3 9	P3 3
P4 7	P4 2

What is the minimum number of resource instances that need to be available for this state to be safe?

Answer:
Need matrix is:
P1 2
P2 1
P3 6
P4 5
We need at least 1 instance. P2 can be satisfied. Available will be 2. Then we can satisfy P1. Then available will be 3. At this point we can not satisfy the others. P4 needs 5. Therefore initially we need to have 3 instances available. Then after P2 and P1 had run, we will have 5 instances available. With that we can satisfy P4. Then we will have 7. We can now satisfy P3. Answer is 3.

Q2. Consider the following situation in a system where there are 5 processes and 3 resource types: A, B, C. No deadlock avoidance or prevention is applied. Initially, there exist 10 A, 6 B, and 4 C in the system (without allocations yet). Is there a deadlock at the moment? Prove your answer.

	Alloc			Request		
	A	B	C	A	B	C
P1	1	0	0	0	0	1
P2	3	0	1	0	4	0
P3	0	2	2	3	1	2
P4	1	0	1	2	3	0
P5	2	1	0	3	3	3

Answer:
Existing = [10, 6, 4]

Available is: [3, 3, 0]
With that we can satisfy P4. Then available becomes: 4, 3, 1.
With that we can satisfy P1. Then available becomes: 5, 3, 1.
No other process can continue running. Hence there is deadlock. P2, P3, P5 are deadlocked.

Q3. Assume we have system that is using single-level paging. Assume page table for a process is always in the memory. a) If a physical memory access takes 200 ns, what is the effective access time to memory (EAT) without TLB? b) Assume we have a TLB used. The TLB search takes 20 ns, no matter it is a hit or miss. If hit rate is 80%, what is the effective access time to memory? c) If two level paging would be used, what would be your answer for questions a) and b)?

Answer:
a) We need 2 physical memory accesses. Hence it is 2 x 200 ns = 400 ns.
b) With TLB, if there is a miss, we need 20 + 200 + 200 = 420 ns.
If there is a hit, we need 20 + 200 = 220 ns.
EAT = 0.8 x 220 + 0.2 x 420 ns = 260 ns.
c) If we have two level pages table used, we need 200x3 = 600 ns without TLB (2 accesses for mapping, 1 access for the data) (a). With TLB, EAT = 0.8 (20+200) + 0.2 (20 + 600) = 300 ns (b).

Q4. Assume a system is using two-level paging, 32-bit virtual addresses and 40-bit physical addresses. A virtual address is divided into three pieces as follows: [`10, 10, 12`]. That means, the first 10 bits are index to the first-level table. Offset is 12 bits. Assume each page table entry (first-level or second-level) is 4 bytes.
a) How many bits in a page table entry are used to store a frame number?
b) How much memory is consumed by first and second level page tables for a process that has 256 KB of virtual memory used, starting at address `0x00000000`.
c) How much memory is consumed by first and second level page tables for a process that has a code segment of 48 KB at virtual address `0x01000000`, a data segment of 600 KB starting at virtual address `0x80000000` and a stack segment of 64 KB starting at virtual address `0xf0000000`. Assume for this question that stack segment also grows upward (towards higher addresses).

Answer:

a) A physical address is 40 bits. 40-12 = 28 bits is a frame number. Hence we need 28 bits for PFN in an entry.
b) A second level table can map 4 MB of virtual memory. 256 KB is much less than 4 MB. Then, we need 1 first level and 1 second level table. Hence we need 2 x 4KB = 8 KB space.
c) A second level table maps 2^22 bytes. That is 0x400000. If we divide 0x1000000 to 0x400000, we get **2^24** / 2^22 = 2^2 = 4. That means the entry 4 in the top level table is used. We need a second level table for mapping code part. 48/4 = 12 entries of that table will be

used. 0x80000000 divided to 2^22 makes: **8x2^28** / 2^22 = 2^9; the data segment starts at 4 MB boundary. Data segment is 600 KB. Hence we need 600/4 = 150 frame numbers to be stored. One second level table is enough. The stack is at address 0xf0000000. This is 15 x 2^28. **15x2^28**/2^22 = 15x2^6 = 15x64. Hence this also starts at a 4 MB boundary. 64/4 = 16. 16 frame numbers neeed to be stored. 1 second level table is enough. In total we need 3 second level tables. 1 first level table. 4 tables in total: 4 x 4KB = 16 KB space is need to store the two level page table for the process.

Q5. Consider the following page reference string of a process.

3 5 4 3 5 6 2 5 2 3 4 2 5 4 2 7 4 7 3

Assume the process has 3 frames that it can use, all empty initially.
a) Assume second chance (i.e., clock) algorithm is used as page replacement algorithm. Assume reference bits (R bits) are cleared after every 5 references (i.e., some time between every 5th and 6th reference). Show the memory state (the pages in memory and their R bit values) after each page reference. Also indicate which reference causes a page fault. Assume after a page fault, when the new page is loaded, its R bit is set to 1. b) Solve the same question for Optimal algorithm.

Answer:
victim pointer (hand) points to the page shown in **bold**.

```
3: 3(1)  f
5: 3(1)  5(1)  f
4: 3(1)  5(1)  4(1)  f
3: 3(1)  5(1)  4(1)
5: 3(1)  5(1)  4(1)

6: 6(1)  5(0)  4(0)  f
2: 6(1)  2(1)  4(0)  f
5: 6(1)  2(1)  5(1)  f
2: 6(1)  2(1)  5(1)
3: 3(1)  2(0)  5(0)  f

4: 3(0)  4(1)  5(0)  f
2: 3(0)  4(1)  2(1)  f
5: 5(1)  4(1)  2(1)  f
4: 5(1)  4(1)  2(1)
2: 5(1)  4(1)  2(1)

7: 5(0)  7(1)  2(0)  f
4: 5(0)  7(1)  4(1)  f
7: 5(0)  7(1)  4(1)
3: 3(1)  7(1)  4(1)  f
```

Q6. Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder (track) 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is as follows:

2069 1212 2296 2800 544 1618 356 1523 4965 3681

Starting from the current head position, what is the total distance (in cylinders/tracks) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms? a) FCFS, b) SCAN, c) C-SCAN.

a) FCFS order used. The tracks are visited in the order shown above.
2150-2069=81.
2069-1212= 857
2296-1212 = 1084
2800-2296 = 504
2800-544 = 2256
1618-544 = 1074
1618-356 = 1262
1523-356 = 1167
4965-1523=3442
4965-3681=1284
Total: 13011.
b) SCAN:
The track numbers in sorted order:
356 544 1212 1523 1618 2069 [2150>>>] 2296 2800 3681 4965
The head will move from track 2150 to the end of the platter (track 4999). On the way, the requests will be served. Then head direction is reversed. Head will move towards track 0. It will go up to the last request in that direction. That request is for a block on track 356. Total head movement is calculated as follows.
(4999-2150)+(4999-356)=7492.
c) C-SCAN: Similar to scan, but when head arrived to track 4999, it will be moved to track 0 without serving any request on the way. Then it is moved towards larger track number again, and requests are served on the way.
356 544 1212 1523 1618 2069 [2150>>>] 2296 2800 3681 4965
Total head movement can be calculated as:
(4999-2150)+(4999-0)+(2069-0)=9917.

Q7. A disk has the following parameters:

Size: 1 TB
RPM: 15000
avg seek time: 3 ms
max transfer rate: 50 MB/s.

a) Assume block size 4 KB. What is the I/O time to read one random block from this disk? How many such transfers can we complete per second? What is the I/O data rate (i.e., throughput).

b) Assume we will read 1000 blocks, that are contiguous on the disk, sequentially. How many such transfers can we complete per second? What is the I/O data rate (i.e., throughput).

Answer:
a) 4KB/50MB/s = 0.078 ms. This is the transfer time for 4 KB. Avg rotational latency is: ((1/ (15000/60)) s) / 2 = 0.002 s. = 2 ms. Then total time to transfer 4 KB is: 2 + 3 + 0.078 = 5.078 ms = T. Number of disk I/Os per second = 1/T = 196. Througput (effective data rate; or rate of I/O): 0.76 MB/s.
b) 1000x4KB / 50 MB = 0.078 seconds = 78 ms. Time for one I/O = 2 + 3 + 78 ms = 83 ms = T. Number of disk I/Os per second = 1/T = 12. Throughput: 46 MB/s.

Q8. Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. Assume in an inode we have 10 direct disk block pointers, one single indirect pointer, one double indirect pointer, and one triple indirect pointer. That means, combined index allocation scheme is used to keep track of the blocks allocated to a file. a) What is the maximum size of a file that can be stored in this file system? b) How many second level index blocks are required for a file X of size 4 GB. c) If nothing, except the inodes, is cached in memory, how many disk block accesses are required to access a byte i) at offset 2^16, ii) at offset 2^21, iii) at offset 2^27 of the file X?

Answer:
a) In an inode we can store: 8 KB / 4 = 2048 pointers. Total file size is: 10+2048+2048^2 + 2048^3 ~= 2^33 blocks. That makes 2^33 x 2^13 = 2^46 Bytes. Therefore max file size supported is ~= 64 TB.
b) A leaf index block can map 2^11 x 2^13 = 2^24 = 16 MB of file data. For 4 GB, we need 2^32 / 2^24 = 2^8 = 256 leaf index blocks. Hence we need to use the single level index structure (one leaf index block), and two level index structures (one first level index block and many second level - leaf - index blocks). We need to use 256 - 1 = 255 second level index blocks. We also have 10 direct pointers. But this will not reduce the second level index block count from 255 to 254. Hence the answer is 255 second-level index blocks.
c)
i) Offset 2^16 is in which block? 2^16 / BS = 2^16/2^13 = 8. It is in the file block 8. Hence the respective block is pointed directly by the inode. Therefore we need to do 1 disk access to retrieve the data block containing the byte at that offset.
ii) offset = 2^21. This is 2 MB. A single index block can map 16 MB of contiguous file data. Therefore, we need to access the index block of the one-level index structure and then the data block. We need 2 disk accesses.

iii) offset = 2^27 = 128 MB. 128 MB / 16 MB = 8. Hence, we need to use the two0-level index structure. So, we need to access the first-level index block of the two-level index, then a second-level index block of this structure and then the data block. We need 3 disk access (3 separate blocks of the disk are accessed).