CS 5348 Operating Systems, Homework #3

1. Consider the page reference sequence for a program P: 0 1 2 3 0 1 2 4 0 1 2 5 3 5 6 3. Load the pages following the page replacement policies given below. Also, compute the number of page faults incurred in each policy. Assume that the working set size is 4.
   (a) LRU
   (b) Clock
   (c) Aging: Assume that the page reference is for the entire system, not from a single program and the memory has only 4 pages. Apply aging policy to the access scenario above and give the aging vectors for all 4 pages. Assume that the system maintains 8 aging bits and before the above accesses the aging vectors for all 8 pages were 0. Also, assume that the background scanning and updating of aging vectors happened after every 2 accesses.

a). There are 8 page faults incurred.

0 1 2 3 | 0
1 2 3 0 | 1
2 3 0 | 1 | 2
3 0 1 2 | 4
0 1 2 4 | 0

1 2 4 0 | 1
2 4 0 | 1 | 2
4 0 1 2 | 5
0 1 2 5 | 3
1 2 5 3 | 5

1 2 3 5 | 6
2 3 5 6 | 3
2 5 6 3

b). There are 11 page faults

0* | 1
0* 1* | 2
0* 1* 2* | 3
0* 1* 2* 3* | 0
0* 1* 2* 3* | 1

0* 1* 2* 3* | 2
0* 1* 2* 3* | 4
4* 1 2 3 | 0
4* 0* 2 3 | 1
4* 0* 1* 3 | 2

4* 0* 1* 2* | 5
5* 0 1 2 | 3
5* 3* 1 2 | 5
5* 3* 1 2 | 6
5* 3* 6* 2 | 3

5* 3* 6* 2

c). There are 8 page faults

Accesses	frame 0	frame 1	frame 2	frame 3
0	1 0000000	0000000	0000000	0000000
1	1 0000000	1 0000000	0000000	0000000
	0 1000000	0 1000000	0000000	0000000
2	0 1000000	0 1000000	1 0000000	0000000
3	0 1000000	0 1000000	1 0000000	1 0000000
	0 0100000	0 0100000	0 1000000	0 1000000
0	1 0100000	0 0100000	0 1000000	0 1000000
1	1 0100000	1 0100000	0 1000000	0 1000000
	0 1010000	0 1010000	0 0100000	0 0100000
2	0 1010000	0 1010000	1 0100000	0 0100000
4	0 1010000	0 1010000	1 0100000	1 0000000
	0 0101000	0 0101000	0 1010000	0 1000000
0	1 0101000	0 0101000	0 1010000	0 1000000
1	1 0101000	1 0101000	0 1010000	0 1000000
	0 1010100	0 1010100	0 0101000	0 0100000
2	0 1010100	0 1010100	1 0101000	0 0100000
5	0 1010100	0 1010100	1 0101000	1 0000000
	0 0101010	0 0101010	0 1010100	0 1000000
3	1 0000000	0 0101010	0 1010100	0 1000000
5	1 0000000	0 0101010	0 1010100	1 0000000
	0 1000000	0 0010101	0 0101010	0 1010000
6	0 1000000	1 0000000	0 0101010	0 1010000
3	1 1000000	1 0000000	0 0101010	0 1010000

2. Consider a Unix-like file system with the I-node structure to index file blocks. The I-node data structure is given in the following: Each I-node contains 7 direct addresses, 2 single indirect address pointers, and one double indirect address pointer. (There is no triple indirect address pointer.) Assume that each address is 32 bits long. Also assume that each logical file block is of size 8KB.
Given a file of size 100MB, is the inode sufficient to address all the blocks of the file.

address = 32 bits = 4 bytes

number of addresses a block can hold = $\frac{8KB}{4B}$ = 2K address

max size = $7 \times 8KB + 2 \times 2K \times 8KB + 1 \times 2K \times 2K \times 8KB$

= 56KB + 32MB + 32GB > 100 MB.

The inode is sufficient to address.

3. Consider a Unix-like file system. It has file block size 4KB. Each inode is of size 128B and its format is as defined in Question 2. The plan is to support the storage of 1M files in the system. The average file size is 16KB.
A disk is used to host the file system, which has 512 sectors on each track and each sector is 512B. The file system starts at track 0. Also, the disk uses the C-Scan algorithm for its arm scheduling. Currently, the disk head is at track 600 with a forward direction.

   (a) What is the rough number of file data blocks to be allocated for the file system?
   (b) How many blocks are needed for the file block map?
   (c) How many inode blocks are needed for the file system?
   (d) How many blocks are needed for the inode map?
   (e) How many blocks total are required in the file system?
   (f) How many tracks are required to host the file system?
   (g) Assume that a file F was nonexisting and a command was just issued to created F in the given file system. F got an inum 1024 and its inode is as shown in the figure below after it is created. Assume that the pointers in the i-node is the block offset from the beginning of the file blocks. Also assume that the OS, upon creating a file, will calculate all the required updates to the disk and issue all the updates to the disk manager (arm scheduler) all together. Which tracks have been visited during the creation of F? Give these tracks in the order they are visited.

file attributes	direct pointer 3500	direct pointer 1000	direct pointer 8000	direct pointer null	direct pointer null	direct pointer null	direct pointer null	single indirect pointer null	single indirect pointer null	double indirect pointer null

   (h) Question (g) can be extended to compute the total disk access time. But I will not give it as a homework question because there are already a lot of questions. You can use any set of disk parameters in the notes to practice access time computation for this one. But it is just for your own practice and it is not a part of this assignment.

a). $1M \times 16\,kB = 16\,GB$

$\frac{16\,GB}{4\,kB} = 4M$

b). $\frac{4M\ bit}{8\ bit/byte} = \frac{2^2 \cdot 2^{20}}{2^3} = 2^{19}\ bytes = 512\,kB$

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

c). $\frac{4\,kB/block}{128\,B} = 2^5 = 32\ inodes/block.$

$\frac{1M\ files}{32\ inodes/block} = \frac{1 \cdot 2^{20}}{2^5} = 2^{15}\ blocks = 32k\ inode\ blocks$

d). $1M\ file \rightarrow 1M\ inodes$

$\frac{1M\ bit}{8\ bit/byte} = \frac{2^{20}}{2^7} = 2^{17}\ bytes = 128\,kB$

$\frac{128\,kB}{4\,kB/block} = 2^5 = 32\ blocks$

e). $1 + 32 + 128 + 32k + 4M = 4227233$ blocks in total

f). Total size = 4227233 blocks × 4kB/block

1 track = 512 sectors × 512 B = 256 kB/track

Number of track = $\frac{\text{Total size}}{\text{size/track}}$ = 66051 tracks

9).

$\frac{256\ KB/track}{4KB/block} = 64\ blocks/track$ $\quad 1+32+128+32k = 32929$

3500 → track number: $\frac{\#36429\ block}{64\ blocks/track} = \#569$ track.

1000 → track number: $\frac{\#32929\ block}{64\ blocks/track} = \#530$ track

8000 → track number: $\frac{\#40929\ block}{64} = \#639$ track

Because of C-scan, the disk head will read from

→ #639 → #530 → #569

4. Consider a 500GB, 7200rpm disk. Each track on the disk has 128 sectors and each sector is 1KB. The disk arm traverses the tracks at 0.1 msec per track and requires a start up time 0.2msec. A file system is built on top of this disk and the size of each file block is 8KB. Note that a file block is always allocated contiguously on the disk. Consider a file of size 80KB. The file is allocated on the disk using indexed allocation scheme and the index table is kept in memory. The file contains 10 file blocks, which are allocated on tracks 105, 120, 112, 101, 113, 106, 116, 108, 111, and 115. Assume that the disk has a 1MB buffer and it uses the Scan algorithm for disk arm scheduling. Compute the time required to access the entire file. (Assume that there are no other disk access requests in the system and the current disk arm is at track 0.)

$\frac{8KB}{1KB} = 8$ sector/block. $\quad$ 7200 rpm => 8.3 msec/track => 4.2 msec rotation delay

$8 \times \frac{1}{128} \times 8.3 = 0.52$ msec/block.

101 − 105 − 106 − 108 − 111 − 112 − 113 − 115 − 116 − 120

total tracks traversed = 120

$T = 120 \times 0.1 + 0.2 + (4.2 + 0.52) \times 10 = 59.4$ msec

5. Consider a log-structured file system with each file block of size 4KB. Now we are updating file F1, and a new segment is started just before this update. The segment starts at block 10000 in the file system. The inode design and F1's inode content are as given in Q3. We add 3 data blocks to F1 at offset 4KB. We also created a new file F2 with 4 data blocks. Now a flush occurred and the current part of the segment is written to disk. The inum for F1 and F2 are 87 and 233, respectively.
   (a) Give the current layout of the new segment (at a high level), indicating the data blocks for each file, the inode blocks, and the imap locations.
   (b) Give the inodes for F1.
   (c) Describe the imap content at a high level, no need to worry about the specific format.

No clue how to solve it.

6. Consider the clock device in Unix system.
   (a) Assume that the current time is March 4, 1971, 10:00:00 a.m. What would be the value in the clock register?
   (b) Assume that current time is 10:22:30. Also, assume that the clock timer unit is 1 second. Consider the following timer requests being issued to the clock earlier. Show the timer data structure.
   (A) Process A issued command "sleep (100)" at time 10:22:00.
   (O) OS set time quantum for current process at time 10:22:10. Time quantum is 30seconds.
   (B) Process B issued command "set_timer (10:25:00)" at time 10:22:20.

a). $(365+31+28+3)\times 86400 + 10\times 3600 = 36928800$

b). [10] [60]—[80]

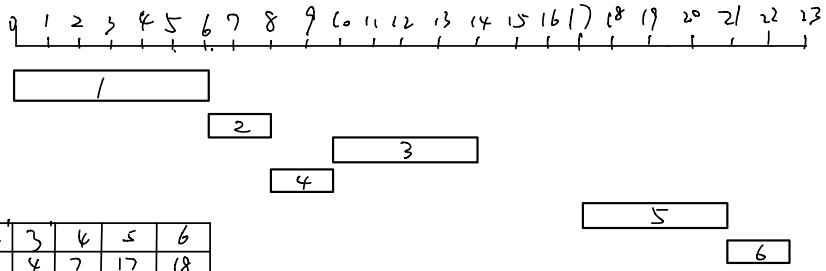
Now 10:22:30
↓10
(O) 10:22:40
↓60
(A) 10:23:40.
↓80
(B) : 10:25:00

7. Consider the following set of jobs.

Job Number	1	2	3	4	5	6
Arrival Time	0	1	4	7	17	18
Service Time	6	2	4	2	4	2

Clearly show the scheduling of these jobs using the following algorithms (refer to the notes).
(a) Shortest job first (non-preemptive)
(b) Highest response ratio first (non-preemptive)
(c) Multilevel feedback queue (number of queue levels = 4, time quantum = 1)

a).

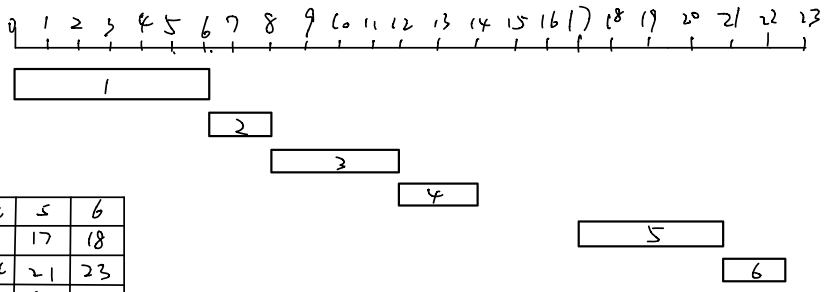


Job#	1	2	3	4	5	6
Arri	0	1	4	7	17	18
Fini	6	8	14	10	21	23
Resp	6	7	10	3	4	5
Serv	6	2	4	2	4	2

Avg response time = 5.83

Avg (response time/service time) = 2

b).

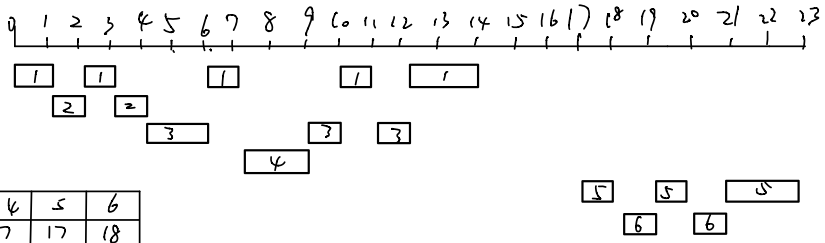


Job#	1	2	3	4	5	6
Arri	0	1	4	7	17	18
Fini	6	8	12	14	21	23
Resp	6	7	8	7	4	5
Serv	6	2	4	2	4	2

Avg response time = 6.17

Avg (reponse time/service time) = 2.25

c).



Job#	1	2	3	4	5	6
Arri	0	1	4	7	17	18
Fini	14	4	12	9	23	21
Resp	14	3	8	2	6	3
Serv	6	2	4	2	4	2

Avg response time = 6

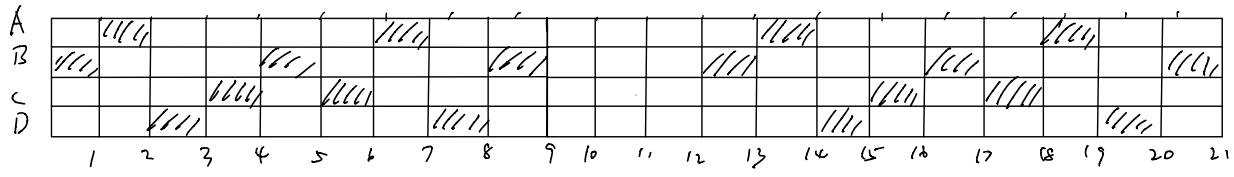
Avg (reponse time/service time) = 1.64

8. Consider a real time system with the periodical jobs specified in the following table. A real time CPU scheduling algorithm, the rate monotonic algorithm, is used. Clearly show the scheduling of the jobs.

Job name	A	B	C	D
Period	6	4	12	6
Service time	1	1	2	1

Sort: B A D C

(4,1) (6,1) (6,1) (12,2)



9. Consider RAID level 4 disk array with a configuration as shown below.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P1
4	5	6	7	P2
8	9	10	11	P3
12	13	14	15	P4
16	17	18	19	P5

(a) If a sequence of read requests retrieve data blocks 1, 6, 8, 9, 13, 15, 16, how many rounds of parallel accesses will need to be performed? Which blocks will be accessed in each round?

(b) If a sequence of write requests access data blocks 1, 3, 4, 6, 7, how many rounds of parallel accesses will need to be performed? Which blocks will be accessed in each round? Give an answer that will require the least number of block accesses.

a) Assume even a sequence is given, OS still able to process without waiting sequentially. then.

3 rounds. Round 1: 1, 6, 15, 8

Round 2: 9, 16

Round 3: 13

b). 5 rounds

Round 1: 1, 3, 4, 6

Round 2: P1

Round 3: P2

Round 4: 7

Round 5: P2