Department of Computer Science and Engineering

IIT Guwahati

CS 343 Operating Systems Quiz 2

Max Marks: 30 **Duration: 75 min**

Solutions to Quiz 2 Questions

Q1) [**3 marks**] Consider a UNIX-like file system that uses blocks of size 1KB and disk addresses of size 4 bytes. For this file system, let the i-node structure contain 10 direct entries, and one each of single, double, and triple indirect pointers. Compute the maximum file size supported by this file system?

Answer)

The i-node holds 10 pointers. The single indirect block holds 256 pointers (1KB/4B=256). The double indirect block is good for 256^2 pointers. The triple indirect block is good for 256^3 pointers. Adding these up, we get a maximum file size of 16,843,018 blocks, which is about 16.06 GB.

3 marks only if all the steps are correct and the final answer is correct (atleast very close).

2 marks if the structure of i-node is correct and the number of pointers is correct.

Q2) **[3 marks**] Assuming that all files and directories are exactly one disk block in size, how many disk reads are required to locate the file named **/usr/bin/firefox/index.html** and read it into memory? Write the steps clearly. Assume that the file descriptor for the root directory is already in memory.

Answer)

Steps for reading file given that file descriptor(fd) for / is already in memory.

Locate fd for usr

Read fd for usr

Read usr

Locate fd for bin

Read fd bin

Read bin

For each level it would be two reads, so total 8 reads.

3 marks for correct steps, including one locate and two reads (one for fd and one for file)

2 marks for missing locate, but mentioning reads in steps correctly. If extra steps included for /

1 mark for atleast identifying steps correctly with one level.

Q3) [**3 marks**] What is the use of **open( )** system call for file reading operation? What would happen without open system call for when an application reads a program? How does the OS handle multiple processes opening a file simultaneously?

Answer)

(i) The open() system call returns a pointer to an entry in the system-wide open file table. If there is no open call, for every file read, the file name has to be specified, which would require locating and reading the i-node (file control block) for every operation. Even with caching, it would be difficult to maintain i-node in memory/cache for subsequent reads. (ii) When another process executes open(), a new entry is added to the per-process open-file table that points to the system-wide open file table and a reference counter (open count value) is updated (incremented) to reflect the number of processes working on the file.

1 mark for each part.

2 marks if any one is not correctly answered, full marks only if all parts are correct.

Q4) [**6 marks**] Consider a file of size 100 blocks being stored in two file systems A and B, that use contiguous allocation and linked allocation, respectively. Compute the number of disk accesses required for the below mentioned operations, for each one of the two different file systems. Assume that the data to be written and the file control blocks are already in memory. For file system A, there is enough room to grow the file only at the end.

a. Adding new block in the middle of the file (between 50th and 51st block)
b. Deleting a block in the middle of the file
c. Adding new block at the beginning of the file

Answer)

For contiguous allocation:

(i) Adding block in the middle requires shifting (reading and writing) 50 blocks, one block further. So that is 100 operations and then one write for new block (101 operations)

(ii) Deleting a block in the middle would require moving shifting 50 blocks (read and write), one block ahead. So 100 operations.

(iii) Adding a new block at the beginning, would require shifting 100 blocks (read and write), one block further. So 200 operations and then one write for new blocks. 201 operations.

For Linked allocation:

(i) A block can be written anywhere, but to 50 blocks to be read and then two writes for pointer update. So 52 operations.

(ii) Deleting a block in the middle would involve traversal across 51 blocks, read the pointer and updating the same before the deleted one. Total 51 reads and 1 write.

(iii) Inserting a block at the beginning would simply involve updating one pointer and writing block, so 2 writes.

1 mark for each answer (no partial marks)

Q5) [**5 marks**] Consider a storage disk with 4 platters (numbered as 0, 1, 2 and 3), 200 cylinders (numbered as 0, 1, . 199), and 256 sectors per track (numbered as 0, 1,.., 255). The following 6 disk requests of the form [**request ID, sector number, cylinder number, platter number**] are received by the disk controller given in the order of arrival time: [**A, 120, 72, 2**], [**B, 180, 134, 1**], [**C, 60, 50, 0**], [**D, 212, 86, 3**], [**E, 56, 126, 2**], [**F, 118, 10, 1**]. Currently the head is positioned at sector number 80 of cylinder 70. The disk is rotating in anticlockwise direction. The average power dissipation in moving the head over 50 cylinders is 35 milliwatts and each reversal of the direction of the head movement dissipates 25 milliwatts.

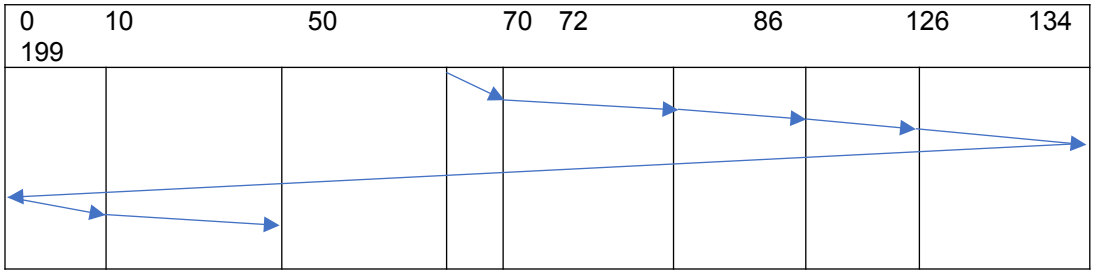
Power dissipation associated with rotational latency and switching of head between different platters is negligible.

Answer)

Request	Service Order	Sector Number	Cylinder number	Platter Number
A	1	120	72	2
B	4	180	134	1
C	6	60	50	0
D	2	212	86	3
E	3	56	126	2
F	5	118	10	1

For power consumption/dissipation, only seek operation matters. Since rotational latency and head switching time is negligible, only cylinder number is needed for the calculation.

a) Head Movement Chart.



Total head movements = (199-70) + (199-0) + (50-0) = 378
Number of reversal of head movement = 2
Total power consumption = (378/50)x35 + (2x25) = 314.6 milliwatts. {312-316 mW is also acceptable}

Marking: head movement chart (1 mark) + calculation (1 mark) + final answer (1 mark)

b)

A new request (Q, r, s, 2) any one of the following cases will give same power value as calculated before.

Case 1: 150<r< 180 and 150<s<180 : r and s can be any value between 150 and 180.

Case 2: 150<r and s<180 : r can be any value larger than 150 and any value of s between 70 and 180 or 0 and 50 will give same power value as calculated before. (s cannot be within 50 and 70)

Marking: 1 mark for r and 1 mark for s.

Q6) [**5 marks**] Consider a single platter storage disk with 98 cylinders (0, 1,…, 97), and 64 sectors per track (0, 1, … , 63). At time T, three disk requests (R1, R2 & R3) of the form [**request id, sector number, cylinder number**] were in the scheduler queue: [**R1, 30, 20**], [**R2, 50, 70**], [**R3, 10, 38**]. Currently the head is positioned (at zero speed) at sector number 50 of cylinder 0. Every seek operation from one cylinder to another involves a mandatory acceleration and deceleration, and an optional coast. When the arm starts from zero speed, it covers 1 track in first 1ms and covers two tracks in next 1ms and there after it can cover 2 more additional tracks in every subsequent millisecond. The coast speed is 6 tracks/ms.

The deceleration also happens similar to the acceleration as described above. Data delivery time of a request is defined as the sum of seek time, rotational latency and transfer time. Assume average rotational latency + transfer time is 2ms per request.

(a) What is the maximum distance (in cylinders) between two disk access requests, if the arm movement should not experience a coast?

(b) How much delivery time will it take to service R1 and then to service R2?

Answer)

98 cylinders (0, 1, , 97), and 64 sectors per track (0, 1, … , 63).
Requests ☾ [R1, 30, 20], [R2, 50, 70], [R3, 10, 38]. The head [H, 50, 0]
1 track/ms, 2 tracks/ms, 4 tracks/ms (acceleration)
6 tracks/ms (coast),
4 tracks/ms, 2 tracks/ms, 1 track/ms (deceleration)
Average rotational latency + transfer time is 2ms per request.

Cylinder #	0	1	3	7	11	13	14
Track /ms		1	2	4	4	2	1
Time		1	2	3	4	5	6

a) The requests can be as wide as 14 cylinders apart for servicing without coast. (2 marks)
b) Delivery time to service R1 and then to service R2.
[R1, 30, 20], [R2, 50, 70]

Cylinder #	0	1	3	7	13	17	19	20
Track /ms		1	2	4	6	4	2	1
Time		1	2	3	4	5	6	7

Cylinder #	20	21	23	27	33	39	45	51	57	63	67	69	70
Track /ms		1	2	4	6	6	6	6	6	6	4	2	1
Time		1	2	3	4	5	6	7	8	9	10	11	12

From cylinder 0 to R1 (cylinder 20) it will take 7 ms to reach and then 2 ms for rotational latency and transfer time.
From R1 (cylinder 20) to R2 (cylinder 70) it will take 12 ms to reach and then 2 ms for rotational latency and transfer time.
Total delivery time: 7 +2 + 12 + 2 = 23 ms.

3 marks if answer is correct with necessary calculations and 2 mark if answer is correct but necessary calculations are missing.

Q7. [**2 marks**] What is memory mapped I/O? Mention why is it used in graphics output devices.

Answer)

In memory mapped I/O, the device data and control registers are mapped to the address space of the processor. The CPU executes I/O requests using standard data transfer instructions to read/write into the mapped locations of memory. In graphics operations, the register space is not enough to hold the screen

contents. So apart from the limited registers for basic I/O operations, memory mapped region is used to hold the content to be displayed on screen. This also avoids I/O instructions for writing millions of bytes.

1 marks for definition of memory mapped I/O and 1 mark for explanation.

Q8. [**3 marks**] Comment on the suitability of (i) polled I/O or interrupt-driven I/O and (ii) buffering and/or caching in an OS for a single user PC, for the following scenarios.

a) A mouse used for interaction with a GUI.
b) A graphics card with direct bus connection, accessible through memory-mapped I/O.
c) A disk drive containing user files.

Answer)

a) Interrupt driven I/O is the most appropriate because the device is a slow one to interact. Buffering may be required when there are high priority tasks that are required to preempt. Caching is inappropriate for interactive applications.

b) Buffering may be needed to control multiple access (to hold the next screen image while displaying the current one). Caching is not necessary due to fast and shared-access nature of the device. Neither Polling nor interrupts are needed for a memory-mapped device.

c) Buffering can be used to hold data in transit from user space to the disk. Caching can be used to improve performance. Interrupt-driven I/O is best for disks that transfer data at slow rates, compared to CPU speed.

For each question part, 2 marks; one each for correctly identifying one function and giving the reason. No marks without justification.