I/O Devices, Disk Scheduling, Files and Directories

Questions:

Q1. A disk has an RPM of 5400, and the seek time per track is 10 ms. There are 500 tracks on the disk with 100 sectors per track. The sectors are numbered 0 to 99 on each track. The SCAN (Elevator) algorithm for disk scheduling. The disk head is initially positioned at track 100, sector 50, moving toward track 0. The following I/O requests need to be scheduled. Calculate the seek time, rotational latency, and total time to reach the sector for each request.

Track 20, sector 25
Track 450, sector 70
Track 150, sector 50
Track 45, sector 65
Track 240, sector 99

Ans. The sequence in which requests will be resolved is as follows.

Track 45 -> Track 20 -> Track 150 -> Track 240 -> Track 450

Seek time = 10 ms
RPS is 5400/60 = 90
Time for one rotation: 1/90 = 0.01111 s = 11.11 ms
Average rotational latency = 0.01111/2 = 0.005 = 5 ms
Rotational Latency per sector = 11.11 / 100 = 0.1111 ms

Either average rotational latency or rotational latency per sector can be used for further calculations.

(i) Track 45, Sector 65
- **Seek Time**: $|100 - 45| \times 10 = 55 \times 10 = 550$ ms
- **Rotational Latency (a)**: $|65 - 50| \times 0.1111 = 15 \times 0.1111 = 1.67$ ms
- **Rotational Latency (b)**: $|65 - 50| \times 5 = 15 \times 5 = 75$ ms
- **Total Time**: 550 + 1.67 = 551.67 ms or 550+75 = 625 ms

(ii) Track 20, Sector 25
- **Seek Time**: $|45 - 20| \times 10 = 25 \times 10 = 250$ ms
- **Rotational Latency (a)**: $|25 - 65| \times 0.1111 = 40 \times 0.1111 = 4.44$ ms
- **Rotational Latency (b)**: $|25 - 65| \times 5 = 200$ ms
- **Total Time**: 250 + 4.44 = 254.44 ms or 250+200 = 450 ms

(iii) Track 150, Sector 50

- **Seek Time**: (|20 - 0| + |150 - 0|) × 10 = 170×10 = 1700 ms
- **Rotational Latency (a)**: |50 - 25| × 0.1111 = 25 × 0.1111 = 2.778 ms
- **Rotational Latency (b)**: |50 - 25| × 5 = 125 ms
- **Total Time**: 1500 + 2.78 = 1502.78 ms or 1500+125 = 1625 ms

(iv) Track 240, Sector 99
- **Seek Time**: |240 - 150| × 10 = 90×10 = 900 ms
- **Rotational Latency (a)**: |99 - 50| × 0.1111 = 49 × 0.1111 = 5.44 ms
- **Rotational Latency (b)**: |99 - 50| × 5 = 245
- **Total Time**: 900 + 5.44 = 905.44 ms or 900 + 245 = 1145 ms

(v) Track 450, Sector 70
- **Seek Time**: |450 - 240| × 10 = 210 × 10 = 2100 ms
- **Rotational Latency (a)**: |70 - 99| × 0.1111 = 29 × 0.1111 = 3.22 ms
- **Rotational Latency (b)**: |70 - 99| × 5 = 145
- **Total Time**: 2100 + 3.22 = 2103.22 ms or 2100 + 145 = 2245 ms

Q2. Consider the following sequence of file operations performed on a file descriptor fd. Assume the initial offset is 0. The file size is 52 bytes.

```
write(fd, buffer, 4);
lseek(fd, 10, SEEK_SET);
write(fd, buffer, 7);
lseek(fd, 5, SEEK_CUR);
read(fd, buffer, 8);
lseek(fd, -10, SEEK_END);write(fd, buffer, 5);
lseek(fd, 10, SEEK_END);
write(fd, buffer, 5);
```

(a) What is the offset of the file after each of the above operations? Show your calculations for each step.
(b) What happens if a read operation is attempted beyond the end of the file in this sequence?

Ans. (a) Offset after each operation:
1. write(fd, buffer, 4) -> **4** (writes 4 bytes, moves forward)
2. lseek(fd, 10, SEEK_SET) -> **10** (sets offset to 10)
3. write(fd, buffer, 7) -> **17** (writes 7 bytes, moves forward)
4. lseek(fd, 5, SEEK_CUR) -> **22** (moves forward by 5)
5. read(fd, buffer, 8) -> **30** (reads 8 bytes, moves forward)
6. lseek(fd, -10, SEEK_END) -> **42** (sets offset to 42)
7. write(fd, buffer, 5) -> **47** (writes 5 bytes, moves forward)

8. lseek(fd, 10, SEEK_END) ->62 (sets offset to 10 past the EOF)
9. write(fd, buffer, 5) ->67 (writes 5 bytes, moves forward)

(b) read() call beyond the EOF returns 0.

**Q3. Using the partial strace output of a program, reconstruct the corresponding C program
that produces this output. Ensure the program handles potential errors during system calls gracefully. Do not use additional libraries except the standard C headers.**

```
openat(AT_FDCWD, "o1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 3
dup2(3, 1)                        = 1
close(3)                          = 0
openat(AT_FDCWD, "o2.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 3
dup2(3, 2)                        = 2
close(3)                          = 0
write(2, "Hello strace! \n", 15) = 15
write(1, "Hello World!\n", 13)   = 13
```

Ans.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
int main() {
        int stdout_fd, stderr_fd;

        stdout_fd = open("o1.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);    - 1
        if (stdout_fd < 0) {
        perror("Error opening o1.txt");
        exit(EXIT_FAILURE);
        }

        if (dup2(stdout_fd, STDOUT_FILENO) < 0) {                          - 2
        perror("Error");
        close(stdout_fd);
        exit(EXIT_FAILURE);
        }

        close(stdout_fd);                                                  - 3

        stderr_fd = open("o2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);    - 4
        if (stderr_fd < 0) {
```

```
        perror("Error opening o2.txt");
        exit(EXIT_FAILURE);
        }

        if (dup2(stderr_fd, STDERR_FILENO) < 0) {                    - 5
        perror("Error");
        close(stderr_fd);
        exit(EXIT_FAILURE);
        }

        close(stderr_fd);                                            - 6

        printf("Hello World!\n");                                    - 7
        fprintf(stderr, "Hello strace! \n");                         - 8

        return 0;
}
```

Q4. A filesystem has a block size of 4 KB, and the Inode size is 128 B. The Inode table start address is 0 B, and the size of each sector on the disk is 512 B. What is the block and disk sector associated with Inode number 8000? To simplify the calculations, you can assume 1 KB = 1000 B instead of 1024 B.

Inode 8000's relative position = 8000*128 = 1024000
Block number = 1024000/4k = 1024000/4000 = 256
Block number if 1KB = 1024 B is used, = 1024000/4*1024 = 1024000/4096 = 250
Sector = (1024000 + 0)/512 = 2000