

BIRZEIT UNIVERSITY

Computer Systems Engineering Department

ENCS 339 Operating Systems

Second Semester, 2020-2021 HW#4 Mass Storage. Due Date: Monday, January 11th , 2021

**Tareq Shannak 1181404**

**Question 1**: Suppose that a disk drive has 2000 cylinders, numbered 0 to 1999. The drive is currently serving a request at cylinder 150, and the previous request met was at cylinder 140. The queue of pending requests, in FIFO order, is

86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

FCFS

SSTF

LOOK

C-SCAN

LCFS

Longest seek time first: LSTF:

Which of the disk-scheduling disciplines mentioned above can cause starvation and which do not. Explain your answer. Describe a way to modify algorithms with starvation to ensure absence of starvation.

FCFS: **150, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130**

**Total Distance (in cylinders)** = 64 + 1384 + 557 + 861 + 826 + 561 + 487 + 728 + 1620

= **7088**

SSTF: **150, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774**

**Total Distance (in cylinders)** = 20 + 44 + 827 + 35 + 74 + 448 + 39 + 241 + 24

= **1752**

**May cause starvation**

LOOK: **150↑, 913, 948, 1022, 1470, 1509, 1750, 1774, 130, 86**

**Total Distance (in cylinders)** = 763 + 35 + 74 + 448 + 39 + 241 + 24 + 1644 + 44

= **3312**

**May cause starvation**

C-SCAN: **150↑, 913, 948, 1022, 1470, 1509, 1750, 1774, 1999, 0, 86,130**

**Total Distance (in cylinders)** = 763 + 35 + 74 + 448 + 39 + 241 + 24 + 225 + 1999 + 86 + 44

= **3978**

**May cause starvation**

LCFS: **150, 130, 1750, 1022, 1509, 948, 1774, 913, 1470, 86**

**Total Distance (in cylinders)** = 20 + 1620 + 728 + 487 + 561 + 826 + 861 + 557 + 1384

= **7044**

LSTF: **150, 1774, 86, 1750, 130, 1509, 913, 1470, 948, 1022**

**Total Distance (in cylinders)** = 1624 + 1688 + 1664 + 1620 + 1379 + 596 + 557 + 522 + 74

= **9724**

**May cause starvation**

**To ensure absence of starvation, we can use aging that increases the priority for older requests.**

**Question 2**: Consider a RAID Level 5 organization comprising nine disks, with the parity for sets of eight blocks on four disks stored on the fifth disk. How many blocks are accessed in order to perform the following?

1. A write of one block of data

Start with read parity block, then we store the old data in the target. If there’s a difference in the two contents(old and new) then we will compute a new parity, finally write the parity and target blocks, so we will have **4 blocks** in total.

1. A write of seven continuous blocks of data.

When we write a 7 continuous blocks, we will write on 4 blocks for parity as there only 4 parity blocks, then the parity for the next four sets is also corresponding parity blocks are written, so we need in total **16 blocks**.

**Question 3**: A UNIX filesystem has 2-­‐KB blocks and 4-­‐byte disk addresses. Each i-­‐node contains 10 direct entries, one single-­indirect entry and one double-­‐indirect entry and 1 triple indirect entry...

1. What is the maximum file size?

Direct Entries Size = 10\*2KB = 20KB

Pointers in each indirect entry = 2KB/4B = 0.5Kpointers

Single Indirect Size (0.5K pointers) =1 \* 0.5K \* 2KB = 1MB

Double Indirect Size (0.5K\*0.5K pointers) = 1\* 0.25M \* 2KB = 0.5GB

Tripe Indirect Size (0.5K\*0.5K\*0.5K pointers) = 1 \* 0.125G \* 2KB = 256GB

Max File Size = 256GB + 0.5GB + 1MB + 20KB

1. If half of all files are exactly 2.5-­‐KB and the other half of all files are exactly 4-­‐KB, what fraction of disk space would be wasted?

For both, they will use 4KB (2\*2K)

4KB file 🡪 No wasted fraction.

2.5KB file 🡪 1.5KB wasted

Fraction Wasted = (1.5KB/2KB) \* 50% = **37.5%**

1. Based on the same condition as in b), does it help to reduce the fraction of wasted disk space if we change the block size to 1-­‐KB? Justify your answer.

**Yes** it reduces the fraction where:

4KB file 🡪 No wasted fraction.

2.5KB file 🡪 0.5KB wasted

Fraction Wasted = (0.5KB/2KB) \* 50% = **12.5%**

Consider a system where free space is kept in a free-space list.

1. Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer.

In order to reconstruct the free list, it would be necessary to perform “garbage collection. “This would entail searching the entire directory structure to determine which pages are already allocated to jobs. Those remaining unallocated pages could be relinked as the free-space list.

1. Consider a file system similar to the one used by UNIX with indexed allocation. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c? Assume that none of the disk blocks is currently being cached.

The free-space list pointer could be stored on the disk, and may be also stored in several places.

**Question 4**: Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions in a table form:

1. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)

|  |  |  |
| --- | --- | --- |
| contiguous | linked | indexed |
| If we want to locate an arbitrary byte at offset X in a file, then X/512 is the virtual block containing the byte.  X%512 is the offset within the block containing the byte.  The logical block being sought is X/512+Y | **Divide the logical and physical address by 512 with X and Y the resulting quotient and remainder respectively. Chase down the L.L (getting X+1 Blocks).**  **Y+1 is the displacement into the last physical blcock.** | **Let’s assume X is the starting point of the index. Then the entry for block 4 (numbered 1 to 4) is the byte at block (X+3)/512 and the offset at 3%512 (3=4-1)** |

1. If we are currently at logical block 4 (the last logical block accessed was block 10) and want to access logical block 25, how many physical blocks must be read from the disk?

|  |  |  |
| --- | --- | --- |
| contiguous | linked | indexed |
| 1 Block | **4 Blocks** | **2 Blocks** |

Good Luck