Problem Set Unit 2

The problem sets ask questions about the important concepts in the reading and lecture materials. Each question should be answered in no more than 150 words.

We are looking for answers in your own words, NOT copies of the explanation given online, in the book, or in lecture material. Please explain your answers as if we are sitting together at a table discussing computer science concepts.

1. (10 pts) Explain why the SSTF scheduling policy tends to favor middle cylinders over the innermost and outermost cylinders.

2. (10 pts) Why is rotational latency usually NOT considered in disk scheduling?

3. (15 pts) What are the tradeoffs with rereading code pages from the file system versus using swap space to store them.

4. (15 pts) Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is:

   2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

   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?

   a. FCFS
   b. SSTF
   c. SCAN
   d. LOOK
   e. C-SCAN
   f. C-LOOK

5. (15 pts) Contrast the performance of the three allocation methods (contiguous, linked, indexed) for both sequential and random file access.

6. (10 pts) Consider a system where free disk space is kept in a free-space list. Suppose the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer.

7. (10 pts) Consider a file system similar to the one used by UNIX (indexed allocation). How many disk operations are required to access the contents of a file at ***/dir1/dir2/file3***? Assume that none of the disk blocks is currently being cached.

8. (15 pts) You are asked to allocate a file according to either a File Allocation Table (FAT) or multi-level indexed allocation (UNIX inode - triply indirect). Assume that the file is 2000 disk blocks long, there are 4 KB per disk block, each pointer in the FAT occupies 4 bytes, the first index block contains 15 entries (of which 12 are direct, and one each is singly indirect, doubly indirect, and triply indirect - see slides or textbook), every other index block contains 15 entries (may be indirect depending on the nesting level), each index block entry takes 4 bytes, and unused index blocks don't count in the total memory cost, though unused entries in partially filled index blocks do count. How many bytes are used to lay out the file when using a
    a. FAT file system?
    b. UNIX-style file system?

    Now suppose that you wish to read the 1099'th block of the file. Assume each of the following counts as one search operation:

    moving from one element to the next in a linked list
    indexing into an index block
    moving from index block to the next

    How many searches are needed to read block 1099 when using the
    c. same FAT file system as above?
    d. same UNIX-style file system as above?