CS471: Operating System Concepts
Module 9: Homework #9
Solution
Points: 20

Question 1 [Points 2] Some systems provide file sharing by maintaining a single copy of a file. Other systems maintain several copies, one for each of the users sharing the file. Discuss the relative merits of each approach.

Ans:
Single copy---less storage; consistent view of the file to all users since all see the same file; strict concurrency control so multiple simultaneous updates will not be permitted; if the server holding the single copy fails, then that file is unavailable to all users; when number of users accessing the file is high, response time will be high to users as they all need to access the same disk;

One copy per user---more storage; there could be issues of inconsistent updates to individual copies by multiple users requiring complex concurrency control protocol; if one copy becomes unavailable, the rest of the users can still access their copies; read requests could be simultaneously serviced by different copies resulting in higher throughput and better response time.

Question 2 [Points 3] The open-file table is used to maintain information about files that are currently open. Should the operating system maintain a separate table for each user or maintain just one table that contains references to files that are currently being accessed by all users? If the same file is being accessed by two different programs or users, should there be separate entries in the open-file table? Explain.

Ans: Just one table that contains references to all files that are currently being accessed. When a file is opened by more than one user, the table should indicate which all users have opened this file. When a user closes a file, that user's entry is removed from that file. When the last user is removed from a file, then that file could be purged from that table. Similarly, when a user wants to open a file, if the file is already in the file table, then the user is entered into that file's entry.

First, this option of having a single table for open files reduces the memory needed to store these tables. In addition, when a file is not needed by any user, then that file can be removed from the cache.

Question [Points 3] 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 the following question: If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

Ans:
Contiguous: One block must be accessed, since the exact disk address of the logical block can be computed from FCB which is already in the memory.

Linked: Since FCB is already in the memory, it would take 4 physical disk block reads; it first reads block 1, then block 2, then block 3, and finally block 4.

Indexed: Assuming that the index table is in the memory, it would take one block read since the index would map the logical block 4 to physical block.

Question 3 [Points 12] Consider a file currently consisting of 100 blocks. Assume that the file control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for **contiguous, linked, and indexed (single-level) allocation strategies**, if, for one block, the following conditions hold. In the contiguous-allocation case, assume that there is no room to grow at the beginning but there is room to grow at the end. Also assume that the block information to be added is stored in memory.

a. The block is added at the beginning.
b. The block is added in the middle.
c. The block is added at the end.
d. The block is removed from the beginning.
e. The block is removed from the middle.
f. The block is removed from the end.

Contiguous:

a. This requires moving the 100 blocks to the right and then writing the new block at the beginning; So 100 reads and 101 write; a total of 201 I/O operations
b. Here, we need to move blocks 51-100 to the right. So 50 reads and 51 writes; a total of 101 I/O operations.
c. There are no block movements; The last block may be written in 1 write at the end of the file; a total of 1 I/O operation.
d. When block 1 is removed: One way: blocks 2-100 are shifted to the left; so 99 reads and 99 writes; a total of 198 I/O operations. Second way: Update FCB in memory to point to block 2 as the first block; 0 I/O operations.
e. When block 50 is removed (it could be 51 also), blocks 51-100 must be shifted to the left; 50 reads and 50 writes; a total of 100 operations. If block 51 is removed, blocks 52-100 must be shifted to the left---49 reads and 49 writes; a total of 98 I/O operations.
f. When block 100 is removed, only FCB in memory is updated; 0 I/O operations

Linked:

a. Write new block; update FCB in main memory; a total of 1 I/O operation
b. We need to add the new block between blocks 50 and 51. So we read 50 blocks, write 1 block, and write 50^{th} block updating its next link; a total of 52 I/O operations.
c. If FCB contains only pointer to first block, then: So we read 100 blocks, write 1 block, and write 100^{th} block updating its next link; a total of 102 I/O operations. If FCB also contains a pointer to the last block, then write new block, read block 100, and update block 100 with a pointer to new block; a total of 3 I/O operations.
d. When block 1 is removed, FCB in memory is updated; zero I/O operations.
e. When block 50 is removed (it could be 51 also), we need to update the link on block 49. For that, we need to read blocks 1-50, and write block 49; a total of 50 reads and 1 writes; a total of 51 I/O operations. If block 51 is removed, we read blocks 1-51, and write block 50; a total 52 I/O operations.

f. When block 100 is removed, blocks 1-99 are read, and block 99 is written. So there will be 100 I/O operations.

Indexed: Assume that the index is already in memory

a. This requires writing one block and updating index in memory; 1 I/O operation
b. This requires writing one block and updating index in memory; 1 I/O operation
c. This requires writing one block and updating index in memory; 1 I/O operation
d. Update index in memory; zero I/O operations.
e. Update index in memory; zero I/O operations
f. Update index in memory; zero I/O operations

Summary

	Contiguous	Linked	Indexed
a	201	1	1
b	101	52	1
c	1	102 (3)	1
d	198 (0)	0	0
e	100 (98)	51 (52)	0
f	0	100	0