Homework 8 - 6/4/2019

Wednesday, May 29, 2019 1:25 PM

Member: Daniel Yan

File System Interface
11.10
11.14
11.17
File System Implementation
12.10
12.15
12.16

Exercise 11.10
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.

- The open-file table (OFT) contains multiple references to open files that are currently being accessed by users and/or processes. When all the processes are complete, only then is the file removed from the OFT.
- There should be separate entries in the OFT for multiple accesses. When no more entries are associated with the file being open, the file can be closed safely.

Exercise 11.14
If the operating system knew that a certain application was going to access file data in a sequential manner, how could it exploit this information to improve performance?

- If the OS knew about the application's access manner, the OS could tell the File System to then prefetch blocks that correspond to the future requests of the blocks in the sequential manner. Through prefetching, the hold up time will be reduced for processes in future requests.

Exercise 11.17
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.

- For systems that maintain a single copy, concurrent updates to a file may result in incorrect information being obtained by the user or the file may be left in an invalid state. However, a single copy will save space comparative to multiple copies, and changes are made separately for users rather than being shared out.
- For systems that maintain several copies for a shared file, the space requirement is greater, however; files are updated without invalid changes which has a higher likelihood of avoiding incorrect information or invalid states from occurring.

Exercise 12.10
Contrast the performance of the three techniques for allocating disk blocks (contiguous, linked, and indexed) for both sequential and random file access.

- Contiguous Sequential
    - Efficient and fast. Files are stored contiguously, so they can be accessed easily sequentially.
- Contiguous Random
    - Decently fast, given adjacent disk block can be determined easily for the targeted position.
- Linked Sequential
    - Decently fast, linked blocks are simply followed in a sequential manner.

- Linked Random
    - Poor, as multiple links of multiple blocks may needed to be traversed before arriving at target.
- Indexed Sequential
    - Efficient and fast, should work akin to Contiguous Sequential as indices are made sequential.
- Indexed Random
    - Decently fast, similar to contiguous random, the index with the target disk block can be determined quite easily,

Exercise 12.15

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:

a. 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.)
b. 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?

- First let's assume L represents the logic block 4.
- Contiguous
    - Divide L by 512 with Q representing the quotient and R representing the remainder.
    A. We should assume that the start block and file length are already in memory. Add Q to L to obtain the physical block number. R is the displacement into that block
    B. We would need to read 1 physical block. As disk head is moved back from 10 to 6. A single read is then used.

- Linked
    - Divide L by 512 with Q representing the quotient and R representing the remainder.
    A. Traverse down the linked list, with the starting block number indicating the start of the file, getting Q+1 blocks. R+1 is the displacement into the tail physical block.
    B. We would need to read 4 physical blocks, first Q, then Q+1 then block 3, then block 4.

- Indexed
    A. We should assume that the entire index is in memory already, and it's directly linked. Place the indexed block into memory. Physical block address would be at a location X, while Y would be the displacement into the desired physical block.
    B. We would need to a single physical block read due to our memory assumption. As the entire index is in memory we know the location of block 4, so it would be a single read.

Exercise 12.16

Consider a file system that uses $inodes$ to represent files. Disk blocks are 8KB in size, and a pointer to a disk block requires 4 bytes. This filesystem has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

- The 12 direct disk blocks would support $12x8KB$
- A single indirect block would support $2048\ or\ 2^{11}$ links (derived from $\frac{8KB}{2} = 2KB$), each of which holds 8 KB, or $2048x8KB$ total.
- Double indirect would be a squared number of links or $2^{22} \Rightarrow 2048x2048x8KB$
- Triple indirect would be a cubed number of links or $2^{33} \Rightarrow 2048x2048x2048x8KB$
- While we could use the direct blocks as well, the triple indirect disk blocks are by far the largest size of storage we would use for this system, as such we would get $2^{33}x\ 2^{13} \Rightarrow 2^{46}bytes = 64\ TB$
- 64 Terabytes would be the maximum size a file in the file system,