HW8

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 system normally contains a system wide open file table and a per-process table. The system wide table contains references to files that are currently being accessed by any user, while the per-process table tracks all files that a process has open. If the same file is to be accessed by two different programs or users a new per-process table would be made for each user and they would share the system wide open file table as long as they had the corresponding same access mode information. File locks mechanisms prevent the users from accessing the data and manipulate at the same time.

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?

The operating system could pre-page the blocks of data that it knew the program would need to access. This would reduce I/O time required when the file data is eventually accessed.

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.

When there are multiple copies of a file and one programmer changes the file, the changes will not appear in the other's copy. With a shared file only one copy exists so all changes and immediately visible to all others.

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 allocated disk are fast to access and don't require many seek in both sequential and random file access. They are simply and fast on access, however, it is difficult to choose free space when disk blocks are allocated and cannot be grown in the future. Contiguously allocated disk also deal with the dynamic storage-allocation problem, in which fragmentation occurs.

Linked allocated disk are fast to sequential access but extremely slow on random file access because the whole linked list need to be traversed to arrive at the random file. This style is also easily susceptible to lost or damaged pointers but solves external fragmentation problem of contiguous block allocation.

Finally Index is the best out of the three options and solves the external fragmentation and size-declaration problem of contiguous. Random access and sequential access are relatively fast compared to the other two allocation options.

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.)**

   Contiguous allocation divides the logical address by the 512 then start b is added to the block to obtain a physical section of unbroken blocks 512 blocks long. b, b+1, b+2, … , b+ 511.

   Linked Divide the logical physical address by 511 with b as the resulting quotient and y the remainder. Chase down the linked list of b + 1 blocks in which y + 1 is the displacement into the last physical block.

   Indexed allocation divide the logical address by 512 with b the number of blocks and y the remaining. Geth the index block into memory and index the physical block as being contained in the index b.

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?**

   Contiguous 1

   Linked 4

   Indexed 2

12.16 Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system 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?

$$8kB = 8192\ bytes.$$

$$8192(block) / 4(pointer) = 2048\ bytes = indirect\ disk\ block$$

$$(12 * 8KB) + (2048 * 8KB) + (2048^2 * 8KB) + (2048^3 * 8KB) = 64\ Terabytes$$