Lecture

- In the lecture on April 27 we were discussing Chapter 10 and 11 (Implementing File System). Block sizes (4096 vs 512 bytes) were discussed based on `fdisk` output for an `ext4` file system. We started with Chapter 12 (Mass Storage Structure).

Exercises

Note, as usual, that you find even more exercises including solutions here :
`http://codex.cs.yale.edu/avi/os-book/OS9/practice-exer-dir/index.html`

Prepare for the Tutorial Session on Wednesday, April 29, 2015:

10.1 Consider a file system where a file can be deleted and its disk space Reclaimed while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?

10.2 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 just maintain one table that contains references to files that are being accessed by all users at the current time? If the same file is being accessed by two different programs or users, should there be separate entries in the open file table?

10.3 What are the advantages and disadvantages of a system providing mandatory locks instead of providing advisory locks whose usage is left to the users' discretion?

10.4 Provide examples of applications that typically access files according to the following methods: i.) Sequential, ii.) Random

10.5 Some systems automatically open a file when it is referenced for the first time, and close the file when the job terminates. Discuss the advantages and disadvantages of this scheme as compared to the more traditional one, where the user has to open and close the file explicitly.

10.6 If the operating system were to know that a certain application is going to access the file data in a sequential manner, how could it exploit this information to improve performance?

10.7 Give an example of an application that could benefit from operating system support for random access to indexed files.

10.8 Discuss the advantages and disadvantages of supporting links to files that cross mount points (that is, the file link refers to a file that is stored in a different volume).

10.9 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.

10.10 Discuss the advantages and disadvantages of associating with remote file systems (stored on file servers) a different set of failure semantics from that associated with local file systems.

10.11 What are the implications of supporting UNIX consistency semantics for shared access for those files that are stored on remote file systems?

11.1 Consider a file system that uses a modified contiguous-allocation scheme with support for extents. A file is a collection of extents, with each extent corresponding to a contiguous set of blocks. A key issue in such systems is the degree of variability in the size of the extents. What are the advantages and disadvantages of the following schemes?

a. All extents are of the same size, and the size is predetermined.

b. Extents can be of any size and are allocated dynamically.

c. Extents can be of a few fixed sizes, and these sizes are predetermined.

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

11.3 What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file?

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

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

b. 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.

c. Suggest a scheme to ensure that the pointer is never lost as a result of memory failure.

11.5 Some file systems allow disk storage to be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks. How could we take advantage of this flexibility to improve performance? What modifications would have to be made to the freespace management scheme in order to support this feature?

11.6 Discuss how performance optimizations for file systems might result in difficulties in maintaining the consistency of the systems in the event of computer crashes.

11.7 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?

11.9 Fragmentation on a storage device could be eliminated by recompaction of the information. Typical disk devices do not have relocation or base registers (such as are used when memory is to be compacted), so how can we relocate files? Give three reasons why recompacting and relocation of files often are avoided.

11.10 Assume that in a particular augmentation of a remote-file-access protocol, each client maintains a name cache that caches translations from file names to corresponding file handles. What issues should we take into account in implementing the name cache?

11.11 Explain why logging metadata updates ensures recovery of a file system after a file system crash.

11.13 Solve the "programming problem" 11.13 from the course book. This will not require any programming. Assume the following content of the two files `file1.txt` and `file3.txt`:

```
user@machine:~/$ cat file1.txt
This is the first example file.

user@machine:~/$ cat file3.txt
This is the file for soft links.
```