Network file systems, such as NFS, use client–server methodology to allow users to access files and directories from remote machines as if they were on local file systems. System calls on the client are translated into network protocols and retranslated into file-system operations on the server. Networking and multiple-client access create challenges in the areas of data consistency and performance.

Due to the fundamental role that file systems play in system operation, their performance and reliability are crucial. Techniques such as log structures and caching help improve performance, while log structures and RAID improve reliability. The WAFL file system is an example of optimization of performance to match a specific I/O load.

**Practice Exercises**

12.1 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.

12.2 What problems could occur if a system allowed a file system to be mounted simultaneously at more than one location?

12.3 Why must the bit map for file allocation be kept on mass storage, rather than in main memory?

12.4 Consider a system that supports the strategies of contiguous, linked, and indexed allocation. What criteria should be used in deciding which strategy is best utilized for a particular file?

12.5 One problem with contiguous allocation is that the user must preallocate enough space for each file. If the file grows to be larger than the space allocated for it, special actions must be taken. One solution to this problem is to define a file structure consisting of an initial contiguous area (of a specified size). If this area is filled, the operating system automatically defines an overflow area that is linked to the initial contiguous area. If the overflow area is filled, another overflow area is allocated. Compare this implementation of a file with the standard contiguous and linked implementations.
12.6 How do caches help improve performance? Why do systems not use more or larger caches if they are so useful?

12.7 Why is it advantageous to the user for an operating system to dynamically allocate its internal tables? What are the penalties to the operating system for doing so?

12.8 Explain how the VFS layer allows an operating system to support multiple types of file systems easily.

Exercises

12.9 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.

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

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

12.12 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.

12.13 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 free-space management scheme in order to support this feature?

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

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?

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

12.18 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?

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

12.20 Consider the following backup scheme:

- **Day 1.** Copy to a backup medium all files from the disk.
- **Day 2.** Copy to another medium all files changed since day 1.
- **Day 3.** Copy to another medium all files changed since day 1.

This differs from the schedule given in Section 12.7.4 by having all subsequent backups copy all files modified since the first full backup. What are the benefits of this system over the one in Section 12.7.4? What are the drawbacks? Are restore operations made easier or more difficult? Explain your answer.

### Programming Problems

The following exercise examines the relationship between files and inodes on a UNIX or Linux system. On these systems, files are represented with inodes. That is, an inode is a file (and vice versa). You can complete this exercise on the Linux virtual machine that is provided with this text. You can also complete the exercise on any Linux, UNIX, or