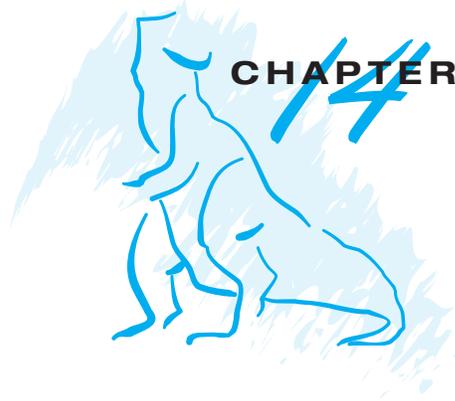# File-System Implementation

Practice Exercises

14.1 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.

Answer:
The results are:

	Contiguous	Linked	Indexed
a.	201	1	1
b.	101	52	1
c.	1	3	1
d.	198	1	0
e.	98	52	0
f.	0	100	0

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

Answer:

In case of a system crash (memory failure), the free-space list would not be lost, as it would be if the bit map had been stored in main memory.

14.3 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?

Answer:

- **Contiguous**—if file is usually accessed sequentially, if file is relatively small.

- **Linked**—if file is large and usually accessed sequentially.

- **Indexed**—if file is large and usually accessed randomly.

14.4 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.

Answer:
This method requires more overhead then the standard contiguous allocation. It requires less overhead than the standard linked allocation.

14.5 How do caches help improve performance? Why do systems not use more or larger caches if they are so useful?

Answer:
Caches allow components of differing speeds to communicate more efficiently by storing data from the slower device, temporarily, in a faster device (the cache). Caches are, almost by definition, more expensive than the devices they are caching for, so increasing the number or size of caches would increase system cost.

14.6 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?

Answer:
Dynamic tables allow more flexibility as a system grows—tables are never exceeded, avoiding artificial use limits. Unfortunately, kernel structures and code are more complicated, so there is more potential for bugs. Dynamic tables use more system resources than static tables, thus potentially taking system resources away from other parts of the system as the system grows.