Ch11 Exercises

11.11 None of the disk-scheduling disciplines, except FCFS, is truly fair (starvation may occur).

a. Explain why this assertion is true.

b. Describe a way to modify algorithms such as SCAN to ensure fairness.

c. Explain why fairness is an important goal in a multi-user systems.

d. Give three or more examples of circumstances in which it is important that the operating system be unfair in serving I/O requests.

Answer:

a. **Explain:**

New requests for the track over which the head currently resides can theoretically arrive as quickly as these requests are being serviced. So a starvation may occur in this situation.

b. **A Way to Modify Algorithms:**

All requests older than some predetermined age could be "forced" to the top of the queue, and an associated bit for each could be set to indicate that no new request could be moved ahead of these requests.

Example: In SCAN scheduling, we determine the next request by looking only at those requests that are in the direction of head movement, and service the closest one next. A simple way to use ageing in this scheme is to track the time each request is in the queue. After a request is some number of milliseconds old, we service it next, no matter what direction the head is moving or where the head is. After servicing this request, we resume SCAN.

c. **Explain:**

Fairness is an extremely important goal in a multi-user systems in order to prevent unusually long response times. As in time shared OS we need everyone should get a fair share of resources they want.

d. **Examples:**

Paging and swapping should take priority over user requests. It may be desirable for other kernel-initiated I/O, such as the writing of file system metadata, to take precedence over user I/O. If the kernel supports real-time process priorities, the I/O requests of those processes should be favored.

The OS should probably service page-fault requests before other I/O requests, service keyboard requests before disk requests, service network requests before disk requests.

11.14 Elementary physics states that when an object is subjected to a constant acceleration a, the relationship between distance d and time t is given by $d = \frac{1}{2}at^2$. Suppose that, during a seek, the disk in Exercise 11.14 accelerates the disk arm at a constant rate for the first half of the seek, then decelerates the disk arm at the same rate for the second half of the seek. Assume that the disk can perform a seek to an adjacent cylinder in 1 millisecond and a full-stroke seek over all 5,000 cylinders in 18 milliseconds.

a. The distance of a seek is the number of cylinders over which the head moves. Explain why the seek time is proportional to the square root of the seek distance.

b. Write an equation for the seek time as a function of the seek distance. This equation should be of the form $t = x + y\sqrt{L}$, where t is the time in milliseconds and L is the seek distance in cylinders.

c. Calculate the total seek time for each of the schedules in Exercise 11.14. Determine which schedule is the fastest (has the smallest total seek time).

d. The **percentage speedup** is the time saved divided by the original time. What is the percentage speedup of the fastest schedule over FCFS?

Answer:

a. **Explain:**

The seek time is proportional to the square root of the seek distance because if $d = \frac{1}{2}at^2$ then $t = \sqrt{\frac{2}{a}d}$.

b. **Equation:**

Using the equation $t = c + d\sqrt{L}$, we can calculate c and d from our two data points (1cyl, 1ms) and (4999cyl, 18ms). Given two unknowns and and data points we can solve for those unknowns. It turns out that $c = 76$ and $d = 24$.

However, we still haven not said what these coefficients really represent in terms of the acceleration, a, of the disk head, the actually seek time t_s and the overhead of doing a seek t_0. The total seek time we will call t_t.

First off, $t_t = t_s + t_0$. Also half the seek time is given by the formula $t = \sqrt{\frac{2}{a}d}$ from above with d as half the cylinder distance. This is because it accelerates half way there and then decelerates the rest of the way. So, $t_s = 2\sqrt{\frac{L}{a}}$.

Plugging this in with our formula for t_t we get: $t_t = t_0 + 2\sqrt{\frac{L}{a}}$. This means that $c = t_0$ and $d = 2\sqrt{\frac{1}{a}}$. We could not solve for a if we wanted.

Note that t_0 is close to 1. If we had made the simple approximation that the (1cyl, 1ms) data point was purely overhead and that the (4999cyl, 18ms) data point was 1ms overhead and 17ms seek time, we would not have been far off.

c. **Total Seek Time:**

Plugging the formula $t_t = 0.76 + 0.24\sqrt{L}$ in for each seek in the schedule we get the following seek times:

FCFS: 64ms, SSTF: 31ms, SCAN: 61ms,

LOOK: 40ms, C-SCAN: 64ms.

The fastest schedule was SSTF.

d. **Percentage Speedup:**

Calculating the percentage speedup of SSTF over FCFS,

we get: $100 \times \frac{64-31}{64} = 52\%$, with respect to the seek time. If we include the overhead of rotational latency and data transfer, the percentage speedup will be less.

Ch13 Exercises

13.14 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?

Answer:

1) If the system knows that the application will access file data sequentially, it can preset the subsequent file blocks of the current file block in advance. In this way, the time that the application has to wait for subsequent file block transfers can be reduced.
2) When a block is accessed, the file system could prefetch the subsequent blocks in anticipation of future requests to these blocks. This prefetching optimization would reduce the waiting time experienced by the process for future requests.

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

Answer:

1) Random access to files comes in two forms: Indexed and Direct. They are both useful when the file is large, and the application needs a record from some arbitrary place in the file.
2) An application that maintains a database of entries could benefit from such support. For instance, if a program is maintaining a student database, then accesses to the database cannot be modeled by any predetermined access pattern. The access to records are random and locating the records would be more efficient if the operating system were to provide some form of tree-based index.

13.16 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.

Answer:

1) **Single Copy:**
   Advantages: the database ensures that all users see the same information. This helps maintain data consistency and reduces the risk of data inconsistencies, such as duplicate or conflicting data.
   Disadvantages: several concurrent updates to a file may result in user obtaining incorrect information, and the file being left in an incorrect state.

2) **Multiple Copies:**
   Advantages: you can easily retrieve or restore if something happens to the original.
   Disadvantages: there is storage waste and the various copies may not be consistent with respect to each other.