OS HW 5

Chapter 11

11.11

Question: 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. Starvation will happen if a request can not be serviced for a long time. In FCFS, every request has the equal priority, they will be serviced in the arrival time order. But in the other disk-scheduling disciplines, as they will choose the request for the track over which the head currently resides to service, so if we continuously give new request, the old request can be delayed to be serviced forever.

b. We can record all the requests' arrival time and add a bit to distinguish the request which has waited for a long time and the others. With this information and the system time, we can get the waiting time for every request, if the waiting time is longer than the threshold we set, these request will be forced to the top of the queue which maintains all the request waiting to be serviced and the relative bit will be set to promise that new request will not go ahead of them again. With this method, if a system use SSTF, it will must organize the rest queue that waiting to be serviced once the old requests are forced to the top.

c. Because if we can't keep the fairness, starvation may come up. It means that if one user continuously give new request that has high priority, other users' low priority requests have to wait extraordinary long time to be serviced even, it is unacceptable in a multi-user systems. So fairness is an important goal in a multi-user systems.

d. There are some examples:

- In order to keep the memory management work properly, the paging and swapping operation should take a high priority which make them serviced before user requests.
- The priority of the foreground operation is higher than that of the background operation
- As the file system work properly need some kernel-initiated I/O, such as the writing of file system meta data, so these kernel-initiated I/O maybe need a higher priority comparing to user I/O operation.
- If there are some important system file need to be accessed kernel, these request maybe need a higher priority comparing to the ordinary I/O requests.

11.13

Question: Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is:

$$2069,\ 1212,\ 2296,\ 2800,\ 544,\ 1618,\ 356,\ 1523,\ 4965,\ 3681$$

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

a. FCFS

b. SCAN

c. C-SCAN

Answer:

a. The **FCFS** schedule is 2150(currently serving), 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681. The calculation of total seek distance is shown below.

$$\begin{gathered}\because (2150-2069)+(2069-1212)+(2296-1212)+(2800-2296)+(2800-544)\\+(1618-544)+(1618-356)+(1523-356)+(4965-1523)+(4965-3681)\\=81+857+1084+504+2256+1074+1262+1167+3442+1284\\=13011\\\therefore \textit{The total seek distance is } 13011.\end{gathered}$$

b. The **SCAN** schedule is 2150(currently serving), 2296, 2800, 3681, 4965, (4999), 2069, 1618, 1523, 1212, 544, 356. (Because the previous request was at cylinder 1805, so the initial status is going in a bigger direction.)

$$\begin{gathered}\because (2296-2150)+(2800-2296)+(3681-2800)+(4965-3681)+(4999-4965)\\+(4999-2069)+(2069-1618)+(1618-1523)+(1523-1212)+(1212-544)+(544-356)\\=146+504+881+1284+34+2930+451+95+311+668+188\\=7492\\\therefore \textit{The total seek distance is } 7492.\end{gathered}$$

c. The **C-SCAN** schedule is 2150(currently serving), 2296, 2800, 3681, 4965, (4999), (0), 356, 544, 1212, 1523, 1618, 2069. (Because the previous request was at cylinder 1805, so the initial status is going in a bigger direction.)

$$\begin{gathered}\because (2296-2150)+(2800-2296)+(3681-2800)+(4965-3681)+(4999-4965)+(4999-0)\\+(356-0)+(544-356)+(1212-544)+(1523-1212)+(1618-1523)+(2069-1618)\\=146+504+881+1284+34+4999+356+188+668+311+95+451\\=9917\\\therefore \textit{The total seek distance is } 9917.\end{gathered}$$

11.14

Question: 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 11.13 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 11.13. 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. In the seeking process, we can divide the distance into two equal part d_1 and d_2, in the d_1 part the disk arm accelerates at a constant rate, then in the d_2 part the disk arm decelerates at the same rate.

So the proof is:

$$\because d_1 = d_2 = \frac{d}{2}$$
$$\because \bar{v}t = d \ \rightarrow \frac{1}{2}(v_0 + v_1)t_1 = d_1$$
$$\because v_1 = a * t_1 \quad v_1 = 0$$
$$\therefore \frac{1}{2}a{t_1}^2 = \frac{d}{2}$$
$$\because \bar{v}t = d \ \rightarrow \frac{1}{2}(v_1 + v_2)t_2 = d_2$$
$$\because v_1 = a * t_1 \quad v_2 = 0$$
$$\therefore \frac{1}{2}a{t_2}^2 = \frac{d}{2}$$
$$\therefore t_1 = t_2 = \sqrt{\frac{d}{a}}$$
$$\therefore t = t_1 + t_2 = 2\sqrt{\frac{d}{a}}$$
$$\because a\ is\ a\ constant$$
$$\therefore t \propto \sqrt{d}$$

b. According to the information 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, the equation is $t = 0.7561 + 0.2439\sqrt{L}$. We can give the following proof.(attention: the unit of t is ms)

$$\begin{cases} 1ms = x + y\sqrt{1} \\ 18ms = x + y\sqrt{4999} \end{cases}$$
$$\therefore \begin{cases} x = 0.7561101813 \approx 0.7561 \\ y = 0.2438898187 \approx 0.2439 \end{cases}$$
$$\therefore t = x + y\sqrt{L} \rightarrow t = 0.7561 + 0.2439\sqrt{L}$$
$$(attention: the\ unit\ of\ t\ is\ ms)$$

c. **The Total Seek Time** for each of the schedules:

- **FCFS : The Total Seek Time** is 90.0249 ms.

$$\because The\ Total\ Seek\ Time = t_1 + t_2 + \ldots + t_n$$
$$\because t_n = 0.7561 + 0.2439\sqrt{L_n}$$
$$\therefore The\ Total\ Seek\ Time\ of\ FCFS = \sum_{n=1}^{10} t_n = \sum_{n=1}^{10} 0.7561 + 0.2439\sqrt{L_n} \approx 90.0249ms$$

- **SCAN : The Total Seek Time** is 68.8491 ms.

$$\because The\ Total\ Seek\ Time = t_1 + t_2 + \ldots + t_n$$
$$\because t_n = 0.7561 + 0.2439\sqrt{L_n}$$
$$\therefore The\ Total\ Seek\ Time\ of\ SCAN = \sum_{n=1}^{10} t_n = \sum_{n=1}^{10} 0.7561 + 0.2439\sqrt{L_n} \approx 68.8491ms$$

- **C-SCAN : The Total Seek Time** is 78.2496 ms.

$$\because The\ Total\ Seek\ Time = t_1 + t_2 + \ldots + t_n$$
$$\because t_n = 0.7561 + 0.2439\sqrt{L_n}$$
$$\therefore The\ Total\ Seek\ Time\ of\ C-SCAN = \sum_{n=1}^{10} t_n = \sum_{n=1}^{10} 0.7561 + 0.2439\sqrt{L_n} \approx 78.2496ms$$

d. **The Percentage Speedup** of the fastest schedule over **FCFS** is 23.5222%.

$$(90.0249ms - 68.8491ms)/90.0249ms * 100\% = 23.5222\%$$

11.15

Question: Suppose that the disk in Exercise 11.14 rotates at 7200 RPM.
a. What is the average rotational latency of this disk drive?
b. What seek distance can be covered in the time that you found for part a?

Answer:

a. As the disk rotates at 7200 RPM, the disk can do 120 rotations per second. So we can indicate that a full rotation will cost $\frac{1}{120}\ s(8.333ms)$. Because the average rotational latency of this disk drive is a half rotation, we can figure out the latency is $\frac{1}{240}\ s(4.167ms)$.

$$\because 7200\ rotaion/min = 120\ rotation/sec$$
$$\therefore \frac{1}{120}\ sec/rotation \approx 8.333ms/rotation$$
$$\because average\ latency = a\ half\ rotation$$
$$\therefore average\ latency = \frac{1}{240}\ sec \approx 4.167ms$$

b. According to the equation got in 11.14, $t = 0.7561 + 0.2439\sqrt{L}$ (the unit of t is ms), we can we can seek over 195 tracks know that during an average rotational latency. And 195 tracks is about 4% of this disk which has 5000 cylinders.

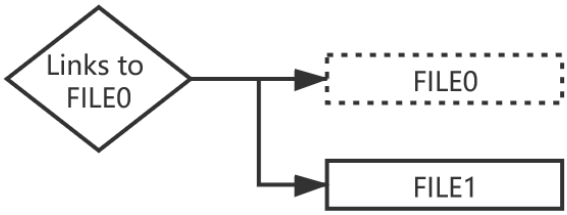
$$\because t = 0.7561 + 0.2439\sqrt{L}$$
$$\because t = average\ latency = \frac{1}{240}\ sec \approx 4.167ms$$
$$\therefore L \approx 195.5372\ tracks > 195\ tracks\ (195/5000 * 100\% \approx 4\%)$$

Chapter 13

13.9

Question: Consider a file system in which 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?

Answer:



As the illustration shown, let **FILE0** be the old file and **FILE1** be the new file. In this situation, if a user want to access **FILE0** with a link, as the real **FILE0** in this path is deleted, the existing link actually point to **FILE1** now. So the user will access **FILE1** instead of get a error notice.

This problem can be solved with a approach that ensure if the file is deleted, the relative links which point to this file should be cleaned as well. To be more specifically, it can be accomplished with a table that recording all the links to **FILE0**(old file), if **FILE0**(old file) is deleted, system will automatically delete all these links. Another Implementation method is that we also keep a table that recording all the links to **FILE0**(old file), but instead of the approach the if **FILE0**(old file) is deleted, system will automatically delete all these links, we stipulate that if the link table is not empty, users can not delete the file, so the user must manually delete all the links, then they can delete **FILE0** legally. The first implementation method is more convenient to users, while the second one can reduce system overhead.

13.13

Question: 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 compared with the more traditional one, where the user has to open and close the file explicitly.

Answer:

a. advantages:

More convenient to the user. If the system can automatically open a file when it is referenced for the first time and close the file when the job terminates, it will relieve the workload of users as they needn't call those file open/close functions to do file operation.

b. disadvantage:

More overhead to the system. Comparing to the traditional scheme the user need to open and close the file explicitly, it will bring more overhead and the system will use more resource as these operation need to be automatically done by the system.

13.16

Question: 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:

a. maintaining a single copy:

First, only keeping a single copy of the file obviously will save storage space. But at the same time, several concurrent modify/update operations to a file may result in user obtaining incorrect information. To be specificly, if user A modified a file, and user B modified the file at the same time, the change made by user A or B may be overwritten by the other one. It can't promise that the file always in a correct status if some conflict operations happen.

b. maintaining several copies:

If the system holding several copies, comparing to keeping a single copy, it will waste storage space. And if the several concurrent modify/update operations happen, the information will not miss. User's change can be recorded. But as there are several copies, if different users do different modifications , it may cause these copies hold different contents. It will cause the sharing property broken.