HW3

Chapter 11

11.11

a.

As we all known, the starvation will happen if a request can't be serviced in time, FCFS haven't the priority of every requests, so every requests will be done in the arrival time order, And in the disk-scheduling, they never get service that the disk-head currently resides the tracks that requested. So if new request happening continuously, the old request never get service.

b.

One way is the search tracks as a loop, just like SCAN algorithm, the other is that we can add a item that priority, a request's priority should be approved first when it waits enough time

c.

Because fairness can avoid the happening of starvation , prevent user to wait a long time.

d.

1. The page fault will be service at first, but the I/O requests will be delayed.
2. The priority of the foreground operation is higher than that of the background operation.
3. In real-time systems, I/O requests from those real-time processes should be served first.

11.13

a.

the order of FCFS is 2150 2069 1212 2296 2800 544 1618 356 1523 4965 3681

the total seek distance is shown below:

$\because (2150-2069)+(2069-1212)+(2296-1212)+(2800-2296)+(2800-544)+(1618-544)+(1618-356)+(4965-1523)+(4965$

b.

the order of SCAN is 2150,2296,2800,3681,4965,4999,2069,1618,1523,1212,544,356

the total seek distance is shown below:

$\because (2296-2150)+(2800-2296)+(3681-2800)+(4965-3681)+(4999-4965)+(4999-2069)+(2069-1618)+(1618-1523)+(1$

c.

the order of C-SCAN is 2150,2296,2800,3681,4965,4999,0,356,544,1212,1523,1618,2069

the total seek distance is shown below:

$\because (2296-2150)+(2800-2296)+(3681-2800)+(4965-3681)+(4999-4965)+(4999-0)+(356-0)+(544-356)+(1212-544)$

11.14

a.

$d=\frac{1}{2}at^2 => t=\sqrt{\frac{2d}{a}}$

b.

For $t=x+y\sqrt{L}$ and ***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.***

we get: $1=x+y\sqrt{1}$

$18=x+y\sqrt{4999}$

Then we get : x=0.7561 and y= 0.2439

so we get: $t=0.7561+0.2439\sqrt{L}$

c.

1. FCFS: the total time = t1+t2+...+tn;

   $t_n=0.7561+0.2439\sqrt{L_n}$

   so the time = 90.0249ms

2. SCAN: the total time = t1+t2+...+tn;

   $t_n=0.7561+0.2439\sqrt{L_n}$

   so the time = 68.8491ms

3. C-SCAN:the total time = t1+t2+...+tn;

   $t_n=0.7561+0.2439\sqrt{L_n}$

   so the time = 78,2469ms

d.

The faster schedule is FCFS , so the Percentage Speedup = $\frac{90.0249-68.8491}{90.0249} * 100\% = 23.5222$

11.15

a.

The average rotational lateny = $\frac{1}{2} * \frac{60s}{7200rpm} = 4.167ms$

b.

$t = x + y\sqrt{L} => L = (\frac{t-x}{y})^2$

$L = 195.58tracks$

Chapter 13

13.9

If file0 is deleted File1 and File2 is the new file, in the situation if a user wanna to access File1 by link, But the File1 is deleted ,the existing link is point to File2, so the user will access File2.

Solution:

1. we delete the link files together when we delete a file, this will need file system contains a structure contains all link files.
2. we could create a counter that count the number of link files, we can delete the file when its counter is 0.

13.13

Pros

It's more convenient for most of users, 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.

Cons

it's more overhead for 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

we set matntaining a single copy as A, the other as B

A.

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.

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

Chapter 14

14.4

Compare with standard contiguous implementation:

with this implementation of a file. the external fragmentation problem of normal contiguous allocation is solved, and as we can allocate overflow area, another contiguous allocation problem which is about the space extension is solved as well. But as these shortages are gone, the internal fragmentation problem appears. Because we allocate an initial contiguous area of a specified size, for the files can not hold all these space, internal fragmentation will come. And if we fix this specified size too large, many space will waste, while the size is too small, many link will be established which will affect the performance of the file system.

Compare with standard linked implementation:

Comparing to the linked allocation, the straight change is that we can reduce the link number so that we can make the performance better in certain situation that we set the specified size bigger than one block. Linked allocation requires that system need to jump with link information every time that we need to read the next block, but in this implementation, only when the block is not in the specified size, we need to do this operation. And if we set the specified size as 1, this implementation transfer to linked allocation.

14.14

Firstly,we get the physical block number and offset number :

$$block\ number\ N = logicoal\ address / 512$$

$$offset\ S = logicoal\ address \% 512$$

a.

Contiguous: physical address= file start address + N * 512 + S

Linked: Jump N times from the start block, get the block address, then add S

Indexed:Find the N-th entry in the index block, then add S

b.

Contiguous: 1, directly access

Linked: 4, sequential access

Indexed:2, one extra access to get address from index