CS 390
Chapter 10 Homework Solutions

10.1 `Is disk scheduling, other than ...`

Yes. Even in a single-user environment, there may be multiple processes competing for access to the disk.

10.2 `Explain why SSTF scheduling tends ...`

The average distance between a cylinder pP and all other cylinders is minimal when p is in the middle of the range of cylinders. If cylinder requests are uniformly distributed[1] then the closest cylinder to a given cylinder is more likely to be in the middle of the disk.

10.3 `Why is rotational latency usually ...`

In an operating systems course, we only study the disk scheduling that can be done by the kernel. With current hard disk technology, the kernel has no way of discovering the rotational angle of the platters (i.e., the sector under the head.) If the kernel could discover the platter angle, it could calculate the additional amount of time required to service each request, and order the requests with the additional latency-wait included.

10.9 `None of the disk scheduling ...`

a. `Explain why this assertion is ...`

In each of the disk scheduling algorithms except FCFS, it is possible for two requests to be serviced in an order that is different than their arrival order. Since the request queue is constantly changing, it is possible for an early request to have its service postponed indefinitely because a later request is serviced before it is.

b. `Describe a way to modify ...`

The traditional way to deal with starvation is ageing. When determining which request to service, the algorithm should take into consideration not only the cylinder of the request, but also the length of time the request has been pending. That is, we age

[1] Is this a reasonable assumption? Somebody should do a capstone on this.

the requests and use this age (along with the cylinder needed by the request) in determining which request to service next.
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 why fairness is an ...`

A good time-sharing system must give each user the illusion that they are the only user of the computer. If resources are not allocated fairly, users will experience delays during some operations.

d. `Give three or more examples ...`

Here are some reasons that I came up with. You can probably think of others.

- The OS should probably service page-fault requests before other I/O requests. (Why?)
- The OS should probably service keyboard requests before before disk requests. (Why?)
- The OS should probably service requests for blocks at the beginning of a magnetic tape before requests for blocks at the end of the tape. (Why?)
- The OS should probably service network requests before disk requests. (Why?)

10.11 `Suppose that a disk drive ...`

The formatting in the text book makes the cylinder numbers ambiguous. I used the following list:
2096, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681.

FCFS	13,011
SSTF	7,532
SCAN	7,492
LOOK	7,424
C-SCAN	9,944
C-LOOK	8,894

10.14 Describe some advantages and disadvantages ...

SSD's are slower and cheaper than primary memory, but faster and more expensive than disk drives, so for both caching and replacement, the result would be a faster and more expensive system. (The latest versions of both Mac OS X and Windows allow the use of an SSD as an extra cache layer between the disk cache (which actually resides in primary memory) and the disk. In Mac OS X, this is called a "fusion drive". In Windows, the technique is called Automated Tiering.

The file caching hierarchy becomes: level 1 disk cache (in primary memory) caches level 2 disk cache (in SSD) caches hard disk.

10.16 Requests are not usually uniformly ...

a. Would any of the scheduling ...

Assuming that the requests are concentrated on cylinders that are close, we would expect SSTF to give the best performance. The idea is that we want to keep the RW head close to these cylinders, since we expect future disk requests to access these cylinders.

b. Propose a disk-scheduling algorithm that ...

We could divide the disk block request list into two queues. One queue would contain I/O requests for the "hot spot" cylinders, and the other queue would contain all other requests. When choosing a request to service, we always service the hot spot queue first, and we service those requests using SSTF. We service this queue first because we expect all the requests in it to be near each other and thus finish quickly. Once the hot spot queue is empty, we service requests from the other queue, using some starvation-free disk scheduling algorithm.

Linux uses a similar disk scheduling algorithm, except that the "hot spot" queue has a timer associated with it. After the timer expires, we service several requests from the other queue before returning to service the hot spot.

10.17 Consider a RAID Level 5 ...

a. A write of one block of data.

This question is confusing because the author's do not distinguish between logical blocks (which the kernel sees) and physical blocks (which are actually transferred by the disk). Assume here that a block is a physical block.

Since RAID 5 uses block-level striping, a write of one physical block of data requires us to read all four physical blocks, compute the new parity for the stripe, and then write 2 blocks back to the disk (one new data block and the new parity) for a total of 5 block accesses. Note, however, that if all the disks are physically different, the reads and writes can proceed in parallel.

b. `A write of seven continuous blocks of data.`

If we assume that the write is "stripe-aligned" (that is begins at the beginning of a physical block), writing seven blocks results in the following block accesses:

- first four blocks: five writes (four data blocks and a parity block),
- last three blocks: one read (for the unchanging block), plus four writes (three new data blocks plus new parity block)

The total is 10 block accesses.

If the write is not stripe-aligned then the write requires 15 disk accesses: A total of 10 writes for the modified data and parity blocks, plus 5 reads for the non-modified blocks on the three stripes which nevertheless must be read in order to calculate parity.

10.18 `Compare the throughput achieved by ...`

a. `A write of one block ...`

In RAID 5, a write of one block requires the system to read 4 data blocks, calculate the parity block, and then write the new data and parity blocks. Note however that all reads proceed in parallel, as do all writes. Thus, the time required is essentially the time needed to read one block and then write one block.

In RAID 1, a write of one block requires two writes - one to the "real" disk, and another to its mirror. Again, these take place in parallel, giving an actual time of one write.

b. `Read operations on multiple contiguous ...`

In RAID 5, any block read requires the system to read all four data blocks in the stripe, since the kernel always performs disk IO in logical block sizes. Since block-striping stores contiguous logical blocks on continuous physical blocks, contiguous reads proceed very quickly. We can essentially read data at four times the speed of a conventional disk when we read the disk sequentially.

In RAID 1, there is no speedup, since we are reading from a single disk. (Traditionally, RAID 1 configurations do not bother to do any error checking. The mirror disk is treated as a backup disk; it is updated on every write, but only read if the first disk fails.)