True/False

1. With, SSD's writing data is straightforward and fast, whereas reading data is complex and slower.

False, it is the opposite. SSD's have complex and slower writes because their memory can't be easily mutated.

2. User applications have to deal with the notion of file blocks, whereas operating systems deal with the finer grained notion of disk sectors.

False, blocks are also an OS concept and are not exposed to users.

3. Directories in UNIX are basically files with pointers to inodes in them.

True, this is basically how directories work.

4. If a given disk read takes 300ms, the greatest source of delay is likely seek time.

No - it's likely queuing time. All of the other delays we talked about relating to disks are bounded in duration at a handful of milliseconds.

Short answer:
5. Indirect blocks look a lot like another data structure we looked at earlier this semester. What is the other data structure? Can you compare the trade-off's you make in either case?

It's very similar to the indirection used in multi level page tables. This also lets you save space (i.e. your inode is smaller in the general case) at the cost of access delay (you have to seek several times). The idea is nearly identical with page tables.

6. What are the major components of disk latency? Explain each one:
Queuing time - How long it spends in the OS queue
Controller - How long it takes to send the message to the controller
Seek - How long the disk head has to move
Rotational - How long the disk rotates for
Transfer - The delay of copying the bytes into memory

Longer answer:
7. How many time is the disk accessed when you type "ls /home/patrickw"

6 times
- Once to read root inode
- Once to read root directory listing
- Once to read home inode
- Once to read home directory listing
- Once to read patrickw inode
- Once to read patrickw directory listing

Assuming the total response time to read an inode or block from disk is 5ms, and all inodes and directories consume one block, how long does this take?

*6 * 5 = 30ms.*

Now say the following is true:
- The root directory listing is cached in memory.
- The disk controller is using a track buffer.
- I-nodes are always stored on the same cylinder as the blocks they refer to.

Assume that data in the track buffer or in memory is “free” (e.g. 0ms) to access. Now much time would it take to read the contents of “/home/patrickw”?

Now it takes 10ms. You don’t have to do the first two seeks, and the 3-4 and 5-6 seeks incur only one seek delay due to track buffering.

8. In class we said that the operating system deals with bad or corrupted sectors. Some disk controllers magically hide failing sectors and re-map to “back-up” locations on disk when a sector fails.

If you had to chose where to lay out these “back-up” sectors on disk - where would you put them? Why?

Should spread them out evenly, so when you replace an arbitrary sector your find one that is close by.

How do you think that the disk controller can check whether a sector has gone bad?

Using a checksum - this can be efficiently checked in hardware during disk access.

Can you think of any drawbacks of hiding errors like this from the operating system?

Excessive sector failures are warning signs that a disk is beginning to fail.