**CS1550 Final exam sample Questions/ Study guide**

These questions are intended to cover the ***kind*** of questions that may be asked about the material we covered. It is not intended as a comprehensive review, and is not representative of the final exam length.

**Paging and Memory Management**

1. Given the following page reference string, the number of page faults using LRU page replacement algorithm for a memory capacity of 3 pages is X, the number of of page faults for the first 8 references in the reference string is Y and the number of of page faults for the last 8 is Z. Reference string: 0, 1, 2, 0, 2, 3, 4, 0, 1, 6, 3, 2, 1, 6, 0, 6. The values of X, Y, Z are:
**A: 13, 6, 7. Using LRU, page faults occur in pages (in what follows, there is a reference string, whether there was a page fault, the state of the main memory frames after the memory was updated, and which page got replaced.** 0 (PF 0--, no replacement), 1 (PF 10-, no replacement), 2 (PF 210, no replacement), 0 (noPF 021, no replacement), 2 (noPF 201, no replacement), 3 (PF, replace 1, 320), 4 (PF, replace 0, 432), 0 (PF, replace 2 043), 1 (PF, replace 3, 104), 6 (PF, replace 4, 610), 3 (PF, replace 0, 361), 2 (PF, replace 1, 236), 1 (PF, replace 6, 123), 6 (PF, replace 3, 612), 0 (PF, replace 4, 061), (6 noPF 021, no replacement)
2. At 1 bit per 8KB chunk, we need X bytes of bitmap per MB of memory.
a. 128 bytes b. 64 bytes c. 32 bytes **d. 16 bytes
1 bit/8KB 🡪 128 bits/1MB 🡪 16bytes/1MB**
3. To guarantee progress, a running process has to be allocated a minimum number of physical frames. This number depends on:
	1. **The degree of multi-programming**
	2. The instruction set of the CPU
	3. The page replacement algorithm
	4. The physical memory size
4. In a virtual memory system with two-level page tables, a running process has at least X page tables loaded in memory.
	1. zero, some memory positions are in cach
	2. **one, which is the first-level page table**
	3. two, namely the first-level page table and a second-level page table
	4. three, namely the first-level page table and two second-level page tables
5. Given a memory system that has 32-bit virtual memory addresses and 4KB pages, how do you extract the page number from the virtual memory address?
	1. (address) >> 20
	2. **(address) >> 12**
	3. address) << 20
	4. (address) << 12

**I/O Questions**

1. Why are SSDs preferred for laptops over hard drives? What are the trade-offs?
**A: SSDs are lighter, faster, offer smaller capacity, less durable (in terms of number of writes) and more expensive. Laptops need to be lighter, everyone needs faster, do not have as many writes as servers, and do not need to be as big as servers.**
2. What changes are needed in the OS file system to optimize the use of SSDs that are not needed for Hard Drives?
**A: SSDs need to be erased before written, avoiding unnecessary writes is better for reliability (in both, avoiding unnecessary writes is better for performance). The TRIM command helps the Flash Translation Layer with garbage collection, erasing blocks ahead of time, not on demand.**
3. A disk has 200 cylinders. The head is currently over cylinder 20 heading outwards (towards higher numbered cylinders). A time t=0 we have a queue of requests for blocks on different cylinders: 25, 36, 28, 124, 56, 18, 178, 236. List the order in which the cylinders will be visited using: **errata: originally, note that the question says 200 cyclinders, but the requests go to 236. Consider request 236 as incorrect, and use 136.**
	1. SCAN **Also know as the elevator algorithm: set a direction, service all requests in that direction, go to end of disk, reverse direction, repeat.** 25, 28, 36, 56, 124, 136, 178, 200, 1, 18.
	2. C-Look **similar to scan, but do not go to the end of the disk if no outstanding requests, and go only in one direction (ie, when reversing, do not stop to read other requests).** 25, 28, 36, 56, 124, 136, 178, 18.
	3. If we assume moving X cylinders required X units of time, then how long will it take to visit all cylinders using SSTF. **The order for SSTF will be** 25, 28, 36, 18, 56, 124, 136, 178. **Total #cylinders traversed (from 20 to 36, to 18, to 178): 17+18+160 = 195. Therefore 195X time units.**
4. Disk scheduling is important since it tries to decrease
**a. the seek time** b. the polling time
c. the rotational delay d. the transfer time
5. Contiguous file systems suffer from similar problems as (in mem mgmt)
a. static partitions **b. dynamic partitions**
c. paging d. none of the above
6. Aging may be one of the techniques through which starvation is avoided in disks. **True**
7. Aging in disks is implemented when aging in CPU scheduling is implemented (ie, they’re dependent on each other). **FALSE**
8. Q: list the three main delays in getting data from  disk, in increasing average delays (or increasing importance, since the more delay, the more important it is) **A: transfer delay, rotational delay, seek delay.**
9. Q: What does the acronym DMA stand for?  Describe what is the advantage of a system that has a DMA over a system that has no DMA
**A: Direct Memory Access. It is a separate piece of hardware that makes transfers from IO devices to memory without the involvement of the CPU, asynchronously. Therefore, can do IO-Memory transfers in parallel with other activities.**
10. Q: If all subcomponents of a system use the bus to get data from memory and the CPU needs data from memory to run the application, why does using a DMA allow for higher performance?
**A: after a chunk of memory was read (say 4KB) into memory (buffer cache), the CPU will not read anything from the IO device until a buffer cache miss was detected.**
11. Q: how does the OS know if it has to read a block from disk, given a logical block address?
**A: a hash table (or some other data structure) is used to keep track of which disk blocks are in the buffer cache; given a disk block logical address, the OS checks this data structure to verify whether the block is still in cache.**
12. Q: given a file pointer, if a program wants to read the next character from the file, how does the OS know where to read from?
13. **A: the file pointer is the address of a data structure that keeps a read/write pointer, which is the offset from the beginning of the file. This pointer is used to compute the block number, similar to page number in virtual addresses when accessing memory. This block number**
14. Q: what is the sequence of software layers that are traversed from the time a user needs to read a disk block until the time the data is available to the user (from library call to the return from the library call).  among the layers are:
a) libraries,   b) page replacement algorithms.  c) ISRs, d) Device-independent OS software, e) data placement algorithms, f) de-framentation software, g) device drivers, h) controllers, i) device itself
**A: libraries  🡪 Device-independent OS software 🡪 device drivers 🡪 controller 🡪 device 🡪 ISR 🡪 device drivers 🡪 Device-independent OS software 🡪 libraries.**
15. Q: Let there be the following requests for data blocks in tracks number 100, 175, 51, 133, 8, 140, 73, and 77 and let the head position be in track number 63.  What is the number of tracks that will be traversed with the following disk scheduling algorithms (answer only those that make sense):
a) FIFO**A: 646 tracks**b) LIFO **A: 623 tracks**c) LRU **typically a page replacement algorithm, not disk scheduling**d) second chance  **but both could be used/adapted to disk scheduling.**e) SCAN **A: 238 tracks**f) C-Look **A: 322 tracks**
16. Q: what is and what is the main shortcomings of the SSTF disk scheduling algorithm?
**A: shortest seek time first algorithm looks at the queue of requests and gives highest priority to the request with the shortest seek time (that is, closest track to the current position of the read-write head). it may starve some requests when requests are arriving dynamically**

**File System Questions**

1. An example disk has 500-byte sectors, and a total capacity of 100MB. All tracks have 100 sectors. It spins at 6000 rpm, and a complete track can be read in 2 revolutions.
	1. How many tracks are on this disk? **A: each track is 50KB (500x100). 100MB/50KB = 2K tracks.**
	2. How fast can data be read off the disk surface (give an upper limit)? **A: Each rotation is 10ms (6000/60). 1 track is read in 2 rotations 🡪 50KB in 20ms 🡪 2K tracks in 20ms x 2K = 40s (disregarding seeks, because there is no information about seek time)**
	3. If we use a file-system with 4-sectors per block, how many bits are needed for a sector pointer? **A: 4-sector blocks 🡪 blocksize = 2KB 🡪 #blocks = 100MB/2KB = 50K blocks 🡪 sector pointer has log2(50K) = 16bits.**
	4. Given your answer to part c), and assuming file names can be up to 200 characters long, then how many direct pointers can be held in a single i-node, if an i-node is exactly 1 file-system block in size? **A: i-node size = 2KB, 200B for name (no other metadata, since it was not mentioned in the question), 1800B left for disk pointers. If only direct pointers are in the i-node, there could be 1800/2=900 direct pointers.**
2. If we use a contiguous file allocation, can we read the entire contents of a file faster/slower than if we use index-nodes?  **A: Faster, given that we don’t have to read the i-node for the file, AND usually contiguous allocations are on the same track (minimizes seek delay/time)**
3. If files are all exactly 10KB in size, and we use 4KB file system blocks, what is the percentage of disk space lost to internal fragmentation. **A:**  **needed 2.5 blocks, internal fragmentation is .5 blocks per file. 0.5/3 = 1/6 of the disk lost to internal fragmentation.**
4. In a log-structured file system, when do you expect a file to be moved on disk (i.e. “moved” from one set of sectors to a new set of sectors)? **A: files are written in memory to start with, and at some point (usually a timeout), moved onto disk, and old blocks invalidated. If blocks are not in memory, that means that they weren’t written or read, typically they’ll remain in the same location as before.**
5. In a log-structured file system, what happens when a file needs to grow (i.e. more disk blocks)? Does the the file system copy the entire file? **A: depending on the size of the update, it may be more efficient to read the entire file on to memory, update it in memory and write it back to disk**
6. What is a good reason to use SSDs as cache for hard drives? **A: performance: SSDs are much faster than HDs.**
7. What is a good reason to avoid using SSDs as cache for memory? **A: performance and endurance: SSDs are much slower than main memory (DRAM) and they only allow for 100K write cycles.**
8. What is a good reason to avoid using SSDs as main memory? **A: performance and endurance: SSDs are much slower than main memory (DRAM) and they only allow for 100K write cycles.**
9. Copy on write is a mechanism implemented in the OS for fast writing of pages from memory onto disk. **FALSE**.
10. Copy on write is a mechanism implemented in the OS to share data and avoid extra movement of data (only copies data upon a write). **TRUE**.
11. Copy on write mechanism can be implemented in the OS by setting the bit in the page table to read only; when a process tries to write, the MMU generates an “exception” (trap into OS), and the OS copies the data. **TRUE**.
12. Directories are typically organized as lists, but can also be organized as trees, usually for faster access. **TRUE**
13. Disk blocks can always be relocated (a dynamic rearrangement of data) at any time with acceptable overhead, since they are small blocks of data.
**FALSE. It depends on the number of disk blocks being moved.**
14. RAIDs allow for both fault tolerance (due to redundancy) and fast access to data on disks, compared with a single disk. **TRUE.**
15. If a process is allocated fewer physical frames than needed to load its working set into memory, thrashing (many many disk operations) may occur. **TRUE.**
16. Both the LRU and the Working Set page replacement algorithms are expensive to implement exactly. **TRUE.**
17. In a disk quota record, there are typically soft block limit and the hard block limits. **TRUE.**
18. Consider an i-node that contains ten direct entries, ten single indirect entries, four double-indirect entries, and two triple-indirect entries. Assume the block size is 512 bytes and that the block number takes four bytes. Compute the maximum file size in bytes.
**A: #pointers per block = 128. Max file size: 10x512 + 10x128x512 + 4x128^2x512 + 2x128^3x512.**

**SHORT ANSWERS**

1. The temporal overhead of log-based file systems (LFS) is mainly due to:
**Having to do compaction and garbage collection when the disk is full (regardless of the utilization of the disk)**
2. The difference between local and global page replacement algorithms is that:
	1. Global can allocate memory from either nodes in the NUMA (ie, Non-Uniform Memory Access)
	2. Local can allocate extra memory when page faults occur, increasing the local allocation
	3. **Local can replace pages from only one process**
	4. Global can replace pages from the OS, if it needs more pages
3. A working set is a set of pages that is required by the process to avoid thrashing (many many disk operations) **TRUE**
4. The working set should reflect the locality of reference for the process, and randomly picking parameters make the working set reflect the set of random memory accesses for a process. **FALSE**

**LONG ANSWER**

1. What is an extent and what is it used for?
**A: Extents are contiguous areas in disk that allow for bigger file sizes, given that instead of a pointer to a single block (e.g., in an i-node), extents point to a set of blocks.**
2. Calculate the expected number of disk accesses for reading an entire 50 KB file, for a 5 MB disk, with 512B disk blocks, given that the directory entry for the current working directory is already in memory, for the following allocation schemes:
	1. Indexed allocation.  **A: 101 disk accesses. Why? First, read index from disk; need to know how many pointers an index node contains. 50MB disk, .5KB blocks 🡪 5MB/.5KB = 2K blocks, need log2(2K) bits = 11 bits. So, #pointers that can be stored in an index block is 512B/11b = 372. Each block is 512B. To read 50KB, we need 100 pointers. So, a single index node will have enough pointers to read the entire 50KB file. Therefore, we need 1 read for the index and 100 reads for the data itself. Total 101 disk accesses.**
	2. Indexed allocation, if the bitmap is not stored on this disk. **A: the answer is the same as above, since the bitmap is only used for when allocating or returning blocks to files. Here, we’re just reading.**
	3. Continuous allocation. **A: assuming the directory entry already has the address of the first block on disk, we will need 100 disk accesses to read the data.**

**Security Questions**

Modern symmetric-key encryption is easily broken by brute-force. **FALSE.**

Describe the “write-up” philosophy for security, and give a short rationale for it. **A: writing up means that you can give information to higher levels of security, and not leak information to lower levels of security.**

Give the formula for the maximum number of attempts for a brute force password attack (ie, trying to guess the password of a user). State your assumptions. **A: assume passwords are less than 8 bytes, the username is already known. Will need to guess all passwords of size 1, 2, 3, …, 8. Each password can be composed of (assume) 70 different characters (e.g., 26 uppercase letters, 26 lowercase, 10 numbers, 18 special characters). So, we’ll need to guess at most 70 possibilities for 1-character password, at most 70^2 possibilities for 2-character password, at most 70^k possibilities for k-character password. Summing all possibilities = 70^9 -1. Big number. About 4e+16, or about 40billion trillion combinations.**

Give an example of covert channels. Give another example of covert channels. **A: 1- increase/decrease CPU usage above below a certain threshold as a means to transmit 0-1 bit values. 2- lock and unlock files as a means to transmit 0-1 bit values.**

Give an example of steganography. **A: replace least significant bits of every pixel in a picture with text (or other smaller sized information/data); this will spread new information out on the picture, without humans (or even computers) noting that any information was removed.**

What are ways to slow down hackers that are trying to log in through bots without knowing the password a priori? **A: several possibilities 1- delay responding exponentially (first few times, delay 100ms, then increase it to 200ms, then 400ms, etc) 2- block IP address for a few minutes after 5 (or 10, or 15) unsuccessful attempts; 3- give requestor a puzzle to solve (like captcha or recaptcha, or a factorization problem); 4- 2-factor authentication.**

Extra questions not considered in the fall 2020 final:

I can fill my backpack with about 20 HDs, each holding 1TB of data. If I can ride my bike from Sennott Square to the Cathedral of Learning in 3 minutes ... what is the bandwidth of my backpack on my bike?
**A: This more of a curiosity, and refers to how fast you can transfer data from one place to another: 20x1TB/180s = 20TB/180s = 111GBps = 888Gbps.**

Why must disk sectors vary in physical length (not storage capacity)
**A: in spinning HDs, the rotation speed is fixed, and the distance from the spindle determines the length of the track and therefore the density of sectors.**

What is sector interleaving? And why wasit invented? Is it still needed?
**A: the ability to skip sectors, to match transfer to rotational speeds, increasing performance. Not really used in the last 20 years, given the advances in technology.**

Q: Describe the organ-pipe distribution and mention why it is a good tool increase the performance of disks.

**A: This block allocation strategy places the most used blocks of data close together in order to reduce the seek time, which is the highest delay in disk performance.  The organ-pipe distribution uses a histogram and allowing the most used blocks of data to be in the same track (say track n/2), the next most used blocks in the next tracks (to the right n/2-1 and to then to the left n/2+1), then n/2+2, n/2-2, …**

**[4 pts]** Give an example where the organ pipe distribution of data increases the seek delays when compared with FIFO data placement algorithms. **There are specific instances such as when the head has to traverse back and forth from one end of the disk to another, when the organ pipe distribution may not be very good.**

1. Intelliseek is a smart algorithm to minimize \_\_\_\_\_. It also minimizes \_\_\_\_\_\_.

**More IO systems**

1. **[5 pts]** Assume that a system has a DMA to perform the block transfers from disk to main memory. Assume that the disk has finished transferring 1K words of data to the controller memory at time *t=0.*  At *t=0* also, the CPU starts a process that will make sequential memory accesses every other bus cycle (that is, the CPU keeps the bus occupied 50% of the time to read a block of 1K words). Assume that the memory bus is shared by the CPU, DMA, disk, and memory. Also assume that only one word can be transferred at a time via the memory bus. How would you design your system to maximize performance for this very particular case?

**All memory accesses of the CPU goes through a cache. In this case, all the cache needs to do is to pre-fetch 1K words of data upon the first memory access to that block of data. Once the cache completes its pre-fetch operation, the bus can be granted to the DMA to complete its operation.**

1. **[3 pts]** What is the sequence of events that happen at the completion of an IO operation? Start at the device.

**Device 🡪 device controller 🡪 interrupt handler 🡪 device driver 🡪 operating system (unblock process) 🡪 user process resumes**

**In the following questions, mark (a) TRUE or (b) FALSE**

1. **[2 pts]** All IO interrupts can be masked by the system (i.e., IO devices can be prevented from interrupting the CPU). **TRUE**
2. An interrupt controller (PIC) arbitrates among interrupts. If an interrupt is being handled:
	1. **[2 pts]** Higher priority interrupts are only able to be executed after the current interrupt is done. **FALSE**
	2. **[2 pts]** Interrupts at the same priority are able execute immediately. **FALSE**
3. **[2 pts]** Interleaving is done in order to accommodate different densities in the disk inner/outer tracks. **FALSE.**

**SHORT-ANSWER QUESTIONS**

1. **[2 pts]** Interrupt handler is not allowed to disable all interrupts, otherwise the process that was interrupted will never execute again and there will be a deadlock. **FALSE. The interrupt handlers are allowed to disable interrupts and may in fact need to disable interrupts. However, the interrupt handlers reside in device drivers or kernel and hence can be ensured to enable interrupts once they are done with their work.**
2. The interrupt service routine is the part of the operating system that handles DMAs. FALSE.
3. Let there be a user whose main file system usage is constantly (a) listening to MP3s (legal, of course!), (b) browsing the web, and (c) sending/receiving emails. Every message and every webpage (all files) get cached in the local disk. Cached files for web browsing are deleted periodically (every T time units).
	1. Create two designs for this user’s file system (the best one you can think of, and another one);
	2. describe what type of file system structure, the type of file allocation strategy, and the disk scheduling algorithm you would use.
	3. Compare your two designs showing advantages of each design (hopefully your “best” design will have more advantages than your “other” design!)

**Dedicate some buffers for caching the MP3 tracks and prefetch them. Rearrangment of disk blocks to allow for whole tracks to be dedicated to MP3s is a good idea. Index allocation due to the frequency of request that are not sequential reads/writes (like the web and email requests). Disk scheduling should be any SCAN or LOOK or elevator algorithms. Just no FIFO.**

**Bonus points for people who say that they’ll have two queues and will put the MP3 requests in higher-priority queue.**

**Alternative design: anything goes :-)**

Public-key cryptography can be used to confidentially exchange information between two parties. Describe how this would be done.

How does PGP (pretty good privacy) work?

What is a one-way hash? Name two well-known algorithms for producing one-way hashes ... and state the size of the resulting hash produced by each (is it a fixed size?).