**T(ch4)**: **1.** In early UNIX systems, executable files (a. out files) began with a very specific magic number, not one chosen at random. These files began with a header, followed by the text and data segments. Why do you think a very specific number was chosen for executable files, whereas other file types had a more-or-less random magic number as the first word? **ANS:** The magic number is used to identifying the file as an executable file. These systems loaded the program directly in memory and began executing at word 0, which was the magic number. To avoid trying to execute the header as code, the magic number was a BRANCH instruction with a target address just below the header. It was possible to read the binary file directly into the new process’ address space and run it at 0, without even knowing how big the header was.

**4.** Systems that support sequential files always have an operation to rewind files. Do systems that support random access files need this too? **ANS:** No. If you want to read the file again, just randomly access byte 0 to go back to the beginning.

**5.** Some operating systems provide a system call rename to give a file a new name. Is there any difference at all between using this call to rename a file and just copying the file to a new file with the new name, followed by deleting the old one? **ANS**: Yes. The rename call does not change the creation time or the time of last modification, but creating a new file causes it to get the current time as both the creation time and the time of last modification. Also, if the disk is full, the copy might fail.

**15.** Consider the i-node structure in a fig. If it contains 10 direct addresses of 4 bytes each and all disk blocks are 1 KB, what is the largest possible file? **ANS:** You have 10 direct addresses to 1KB disk blocks, so that’s 10KB to start with. Then your indirect block points to a 1024B disk block filled with 4byte pointers, (1025/4) giving a 256 additional blocks. Add them together =266KB.

**17.** Two computer science students, Carolyn and Elinor, are discussing inodes. Carolyn maintains that memories have gotten so large and so cheap that when a file is opened, it is simpler and faster just to fetch a new copy of the i-node into the inode table, rather than search the entire table to see if it is already there. Elinor disagrees. Who is right? **ANS:** Elinor is right. Having two copies of the i-node in the table at the same time is a disaster, unless both are read only. The worst case is when both are being updated simultaneously. When the i-nodes are written back to the disk, whichever one gets written last will erase the changes made by the other one, and disk blocks will be lost.

**21.** What would happen if the bitmap or free list containing the information about free disk blocks was completely lost due to a crash? Is there any way to recover from this disaster, or is it bye-bye disk? Discuss your answers for UNIX and the FAT-1 6 file system separately. **ANS:** The recovery algorithm is to make a list of all the blocks in all the files and take the complement as the new free list. In UNIX this can be done by scanning all the i-nodes. In the FAT file system, the problem cannot occur because there is no free list. But even if there were, all that would have to be done to recover it is to scan the FAT looking for free entries.

**33.** How many disk operations are needed to fetch the i-node for the file /usr/ast/courses/os/handout.t? Assume that the i-node for the root directory is in memory, but nothing else along the path is in memory. Also assume that all directories fit in one disk block.

**ANS:** The following disk reads are needed:

directory for /

i-node for /usr

directory for /usr

i-node for /usr/ast

directory for /usr/ast

i-node for /usr/ast/courses

directory for /usr/ast/courses

i-node for /usr/ast/courses/os

directory for /usr/ast/courses/os

i-node for /usr/ast/courses/os/handout.t

In total, 10 disk reads are required.

**34.** In many UNIX systems, the i-nodes are kept at the start of the disk. An alternative design is to allocate an i-node when a file is created and put the i-node at the start of the first block of the file. Discuss the pros and cons of this alternative. **ANS:** PROS: First, no disk space is wasted on unused i-nodes. Second, it is not possible to run out of i-nodes. Third, less disk movement is needed since the i-node and the initial data can be read in one operation. Some cons are as follows. First, directory entries will now need a 32-bit disk address instead of a 16-bit i-node number. Second, an entire disk will be used even for files which contain no data (empty files, device files). Third, file system integrity checks will be slower because of the know the size of inodes, one can easily find which every node in memory by looking at the address calculated by (inode number\* size of inodes). If the inodes are scattered it becomes difficult to search for them.. Fourth, files whose size has been carefully designed to fit the block size will no longer fit the block size due to the i-node, messing up performance.

**T(ch5): 13.** Why are output files for the printer normally spooled on disk before being printed? **ANS:** If the printer were assigned as soon as the output appeared, a process could tie up the printer by printing a few characters and then going to sleep for a year.

**22**. A disk manufacturer has two 5.25-inch disks that each have 10,000 cylinders. The newer one has double the linear recording density of the older one. Which disk properties are better on the newer drive and which are the same? **ANS:** The drive capacity and transfer rates are doubled. The seek time and average rotational delay are the same.

**30.** A system simulates multiple clocks by chaining all pending clock requests together as shown in Fig. 5-34. Suppose the current time is 5000 and there are pending clock requests for time 5008, 5012, 5015, 5029, and 5037. Show the values of Clock header, Current time, and Next signal at times 5000, 5005, and 5013. Suppose a new (pending) signal arrives at time 5017 for 5033. Show the values of Clock header, Current time and Next signal at time 5023. **ANS:** At time 5000: Current time = 5000; Next Signal = 8; Header → 8 → 4 → 3 → 14 → 8. At time 5005: Current time = 5005; Next Signal = 3; Header → 3 → 4 → 3 → 14 → 8. At time 5013: Current time = 5013; Next Signal = 2; Header 2 → 14 → 8. At time 5023: Current time = 5023; Next Signal = 6; Header → 6 → 4 → 4.

**T(ch10): 4**. Why does Linux distinguish between standard output and standard error, when both default to the terminal? **ANS:** They are separate, so standard output can be redirected without affecting standard error. In a pipeline, standard output may go to another process, but standard error still writes on the terminal.

**9.** Does it make sense to take away a process' memory when it enters zombie state? Why or why not? **ANS:** Yes. It cannot run anymore, so the earlier its memory goes back on the free list, the better.

**14.** In every process' entry in the task structure, the PID of the parent is stored. Why? **ANS:** When the process exits, the parent will be given the exit status of its child. The PID is needed to be able to identify the parent so the exit status can be transferred to the correct process.

**17.** When booting Linux (or most other operating systems for that matter), the bootstrap loader in sector 0 of the disk first loads a boot program which then loads the operating system. Why is this extra step necessary? Surely it would be simpler to have the bootstrap loader in sector 0 just load the operating system directly. **ANS:** The program loaded from block 0 is a maximum of 512 bytes long, so it cannot be very complicated. Instead, the block 0 loader just fetches another loader from a fixed location on the disk partition. This program can be much longer and system specific so that it can find and load the OS. We can also have run a program to pick which OS(if multiple) to load. We can also load an OS from a network.

**20.** In Linux, the data and stack segments are paged and swapped to a scratch copy kept on a special paging disk or partition, but the text segment uses the executable binary file instead. Why? **ANS:** The text segment cannot change, so it never has to be paged out. If its frames are needed, they can just be abandoned. The pages can always be retrieved from the file system. The data segment must not be paged back to the executable file, because it is likely that it has changed since being brought in. Paging it back would ruin the executable file. The stack segment is not even present in the executable file.

**(~T)(4):** When there is a lot of swapping activity, computer efficiency drops because writing to disk is much, MUCH slower than writing to RAM. We sometimes simulate file systems [and thereby make them very fast] by holding them in RAM. Why not put the swap area in RAM as well? **ANS:** Because that would be redundant. We swap into the HDD to free up room in RAM. Assuming “swap area” is the same as “swap space”, then it makes no sense at all (by definition) to put swap space in RAM.

**24.** Can a page fault ever lead to the faulting process being terminated? If so, give an example. If not, why not? **ANS:** It is possible. For example, when the stack grows beyond the bottom page, a page fault occurs and the operating system normally assigns the next- lowest page to it. However, it the stack has bumped into the data segment, the next page cannot be allocated to the stack, so the process must be terminated because it has run out of virtual address space. Also, even if there is another page available in virtual memory, the paging area of the disk might be full, making it impossible to allocate backing store for the new page, which would also terminate the process. If you attempt to follow a pointer to an address in the kernel's virtual address space [while in user mode], this will generate a page fault (since you don't have access to the kernel data) and you will be summarily terminated with extreme prejudice (since you're not allowed to look at this privileged data).

**25.** Is it possible that with the buddy system of memory management it ever occurs that two adjacent blocks of free memory of the same size co-exist without being merged into one block? If so, explain how. If not, show that it is impossible. **ANS:** It is possible if the two blocks are not buddies. Two new requests come in for eight pages each. At this point the bottom 32 pages of memory are owned by four different users, each with eight pages. Now users 1 and 2 release their pages, but users 0 and 3 hold theirs. This yields a situation with eight pages used, eight pages free, eight pages free, and eight pages used. We have two adjacent blocks of equal size that cannot be merged because they are not buddies.

**27**. Give two examples of the advantages of relative path names over absolute ones. **ANS:** Opening a file by a path relative to the working directory is usually more convenient for the programmer or user, since a shorter path name is needed. It is also usually much simpler and requires fewer disk accesses.

**31.** If a Linux file has protection mode 755 (octal), what can the owner, the owner's group, and everyone else do to the file? **ANS:** In binary: 111 101 101. The owner can read, write, and execute it, and everyone else (including the owner’s group) can just read and execute it, but not write it.

**33.** Both Fred and Lisa have access to the file x in their respective directories after linking. Is this access completely symmetrical in the sense that anything one of them can do with it the other one can too? **ANS:** No. The file still has only one owner. If, for example, only the owner can write on the file, the other party cannot do so. Linking a file into your directory does not suddenly give you any rights you did not have before. It just creates a new path for accessing the file.

**37.** When an i-node is read in from the disk during the process of opening a file, it is put into an i-node table in memory. This table has some fields that are not present on the disk. One of them is a counter that keeps track of the number of times the i-node has been opened. Why is this field needed? **ANS:** When a file is closed, the counter of its i-node in memory is decremented. If it is greater than zero, the i-node cannot be removed from the table because the file is still open in some process. Only when the counter hits zero can the i-node be removed. Without the reference count, the system would not know when to remove the i-node from the table.

OTHER(old final-ish):

**1)** What attributes of a disk would change if you doubled its RPM? Attributes such as disk capacity, seek time, transfer rate, latency? **ANS:** disk capacity and seek time don’t change but, transfer rate(higher) and latency(higher) change.

2) Hit rate for TLB: (h = hit rate) lookup time in TLB \* h + (1-h) \* reading from page table = average overhead

**3)a)** What would change in the inode for a file if you created a hard link?

**ANS:** A hardlink will create another file(not inode) that points to the same inode. The inode’s link count is then incremented.

**b**) What would change if you created a soft link?

**ANS:** A new inode is created which points to a data block containing the path to the file being linked to. When you open() your soft link, it redirects to the file pointed to by the path.

**c)** Would a new inode be created during either a soft or hard link?

**ANS:** It would not be created for a hard link, but it would be created for a softlink.

**8)a)** What is memory-mapped I/O?

**ANS:** Introduced with the PDP11 godlike computer, each control register is assigned a unique memory address to which no memory is assigned. These addresses are usually at the top of the address space. These registers are just variables in memory and can be addressed in C, thus requiring no assembly code to do reads and writes.

**b)** Why is programming simpler on memory-mapped I/O?

**ANS:** The registers are just variables in memory and can be addressed in C, thus requiring no assembly code to do reads and writes. An IO Driver can be written entirely in C.

**c)** Why does memory-mapped I/O lead to complications with modern architecture?

**ANS:** On older systems, the memory and CPU shared a bus. Today, the I/O devices can’t see the memory references on the high speed bus and would be left in the dark without a special device called a bridge.

**9)** Have a good understanding about your Shell project (ours was p2 & p4) and why you use commands such as dup2/pipe closes etc..

**ANS:** Dup2 duplicates one file descriptor over another. (For redirection).

-We need to close all the pipe ends in all of the places because once the parent forks, it AND its children has open file descriptors to the pipe. If the process tries to read from the pipe, and there’s nothing to read, the pipe will return EOF.

**12)a)** Why are we switching to GPT standard?

**ANS:** MBR is stored in the first 512 bytes of the partition. It contains machine instructions for booting the computer. It did not, however, have the ability to navigate partitions or file systems. It contained code to load a second-stage bootloader, which could. MBR is limited to disks up to 2TB and only 4 primary partitions.

GPT, GUID Partition Table, however, replaces MBR. It starts by giving each partition a Globally Unique IDentifier for each partition. In theory, every GPT partition ever has a unique identifier, due to how long the string is. Drives can be super huge and the number of partitions can be much, much higher. Windows allows for 128 partitions. Is backwards compatible so it can be used on MBR machines.

**E(ch4) 2.**mv ~/.login ~/save/moved.login

**a)** What (if anything) gets changed in the inode for your home directory ~ ?**ANS:** Entry size is changed to turn the file into padding, so its entry is skipped.

**b)** What (if anything) gets changed in the data pointed to by inode for your home directory ~? **ANS**: In the data blocks the entry for the file is “moved”.

**c)** What (if anything) gets changed in the inode for your .login file? **ANS:** CTime is modified goes up and then down.

**d)** Does the data pointed to by the inode for your .login file have to be copied to a new set of blocks? **ANS:** No, the file gets moved but the inode points to the same data blocks, just the file itself is at a new location.

**e)** Suppose, instead, you were to move the file to a directory on a different disk drive (or a different partition), e.g. mv ~/.login /tmp/moved.login Discuss how the answers you gave above would be different. **ANS:** The data block is copied and new inode is created.

**3.** As you know, UNIX maintains a block buffer cache in RAM. However, writes to blocks that contain directory information are immediately written back to disks. Why is the efficiency of having a buffer cache forsaken in this instance? **ANS:** Buffer can be used for re-finding a file that was recently used, if it is immediately written back there is no need to store it in cache.

**4.** Suppose that someone notice that disk reads were proceeding faster than disk writes on your computer system. What would you say? **ANS:** a buffer cache stores recently/frequently used read disk blocks to RAM which is much faster than accessing the disk. Writes have to access the disk regardless making these much slower than reads.

**5.**Suppose that someone noticed that disk writes were proceeding faster than disk reads on your computer system. What would you say? **ANS**: If the filesystem is a log-structured file, this produces slower reads than writes.

**6**. List two different ways an operating system could know which program to run. **ANS:** Window: check file name extension, Unix: check magic number first two byte.

**7**. Pits and lands on a mass-produced commercial CD are equally reflective, and yet your CD-player detects differences in reflected intensity.Explain. **ANS**: Pits are 20 microns closer which is ¼ of the wavelength of infrared laser beam. Batch of waves in lock step that hit pits travel ½ wavelength further than the surrounding surface. Transitions from pits to lands encode the data where you can tell when you move from one height to the next signifying no change is a 0 and a change being a 1.

**8**. Discuss the differences between mass-produced commercial CDs, CD-Rs, and CD-RWs. **ANS:** CD’s had to be mastered on hugely expensive equipment CDR’s have a thin chemical layer (near the label, below the reflective coating) which is transparent until heated by a laser strong enough to “write” on the disk, the heated dye then becomes permanently dark(write once medium). CDRW’s fancy alloy is used for reflective surface and no dye is involved. In its natural state the alloy reflects very well. However if you blast it with a high powered laser you can heat it up enough so that it reflects poorly. So the untouched light spot areas are the lands and the amorphous dark spots are the pits. Cool thing about this alloy is that if you heat it up less violently the structure will reform and become highly reflective again allowing multiple writes.

**9.** What is the motivation for using a log-based filesystem? **ANS**: We do much more writing than reading, so clumping our writes into segments to write at once at 100% write efficiency, inodes are kept separate from the log so that you don’t search the entire log for up to date inode data. Saves a not of overhead. Research shows we do more small writes than reads.

**10**. Why is GPT replacing the older IDE partitioning scheme? **ANS**: GPT REPLACE IDE GPT allows for more partitions and in a more flexible structure.

**11**. Describe a few of the capabilities that UEFI(Unified Extensible Firmware Interface) has that the old BIOS(Basic Input/Output System)does not.

**ANS: a)** includes fancy graphics support and other device drivers that awakening OS can use until it loads its own device drivers.

**b)** ‘Secure Boot’ capability which can ensure that only ‘acceptable’ OS loaders are allowed to function.

**c)** Can be queried over the network after power-on, even if the system has not had an OS installed on it, allowing remote diagnostics and recovery services.

**12.** What is meant by ‘virtual layer’ on a flash drive, what is it’s purpose? **ANS:** In order to even out the wear on the set of blocks.

**13.** What is wear-leveling? **ANS:** is a technique for prolonging the service life of some kinds of erasable computer storage media, such as flash memory used in solid-state drives (SSDs) and USB flash drives.

**14.** What is the difference between nand-based and nor-based flash technology? **ANS:** Nor is used for things like BIOS since only small amounts are required and is byte addressable meaning you can write a single byte w/o affecting any other bytes. Nand is page addressable and is therefore a natural fit for file systems. Can easily turn any particular 1-bit into 0-bit, but only way to turn a 0 into 1 is to erase an entire page of memory, which turns the entire page into 1’s.

**15.** A flash drive is much faster than an IDE drive; swapping processes in and out when the CPU is overcrowded can drastically slow down a system. This would seem to make a flash drive ideal for hosting a swap partition. However, this is never done Why is it a bad idea to use a flash drive as a swap partition? **ANS:** flash technology typically stops being reliable after 100,000 erasures. This is a problem for normal file systems where some blocks get used over and over such as when doing swap partitions. This would tend to “wear” out parts of the flash drive earlier than other parts. This would be due to the fact that some pages which contain system binaries and libraries stored might not ever change; rendering the rest of the blocks of memory used a lot more.

**16.A)**make a hard link, ln p2.c hard.c What fields in the inode for the original p2.c are changed? **ANS:** The number of file Nlink count is increasing one

**B)** soft link, ln -s p2.c soft.c What fields in the inode for the original p2.c are changed? **ANS:** No Fields are changed because you simply just create another Inode containing the same data, not modifying or copying anything in the original.

**c)** Was an additional new inode needed to create either hard.c or soft.c? **ANS:** Yes when you make a soft link a new inode is created containing all the original data. For hard links no, inodes not created.

**17)** Assume user kjones owns a file whose path is /tmp/js/kj with permissions -rwxr—r--, but user jsmith owns the directory /tmp/js, with permissions drwxr-x—x.

**a)** If the file and the directory are in the carroll group, who can read /tmp/js/kj? **ANS:** everyone

**b)** the command “ls /tmp/js” will succeed for which users? **ANS:** Group and user/owner

**b)** Which users can change the contents for /tmp/js/kj? **ANS:** User/owner of file

**c)** Which users can delete /tmp/js/kj? **ANS:** Owner of the directory

**18)** One of the first timesharing machines, the PDP-1, had a memory of 4k 18-bit words. It held one process at a time in memory. When the scheduler decided to run another process, the process in memory was written to a paging drum, with 4k 18-bit words around the circumference of the drum. The drum could start writing (or reading) at any word, rather than only at word 0. Why do you suppose this drum was chosen? **ANS:** The rotational delay is the time taken for bringing the required sector to be read, under the read/write head. The drum could read or write any part of the drum. It does not need a specific sector under the rw-head. Thus PDP-1 eliminated any rotational delay and thus saved a saved a rotational each time a read and write was performed on the drum.

E(ch5) **1.** For a hard disk that spins at 7200 RPM, what is the average rotational latency? FORMULA: (1/(RPM/60))\*.5\*1000 (= 4.17 ms)

**2.** Explain the problems an ‘old’ CD-ROM drive (one that doesn’t know about the CD-R format) might have when reading a CD-R disk. **ANS:** Old cd drives assume the cd-r is only written to once, New formats can continue writing incrementally post previously written data, but old drives do not know to check past the first set of data for new data.

**3.** What is a soft timer? **ANS:** Is a time-ordered list of scheduled events that is checked every time the kernel exits kernel mode, This gives a compromise between polling and interrupts.

**4**. How does pipelining and superscalar architecture complicate the processing of interrupts? **ANS:** With pipelining, the fetched and decoded instructions will be different, and with superscalar architecture the instructions may be executed out of order, so preserving and restoring the state of the process may be very complicated when handling an interrupt.

**5. (a)** What is memory-mapped I/O? **ANS:** Part of the address space does not refer to bytes of RAM in the usually way, but instead is mapped to various controller-card registers. In this scheme writing to some “high” RAM memory address effectively routes data to a register on some controller.

**(b)** How does today’s advanced bus architecture complicate memory mapped I/O? **ANS**: On modern systems the CPU and memory have their own high speed bus and I/O devices cannot ‘see’ the memory references whizzing by on the high-bandwidth bus, and would be left in the dark unless extra hardware allow them to ‘see’ the relevant part of memory.

**6.** What is the difference between scan codes and ASCII codes? **ANS:** The byte of information that is communicated across the port is NOT and ASCII code, but rather a different code(called a scan code) that specifies a physical key. For example, on the standard 102- key US keyboard, 7 bts are used to encode which of the keys changed state, and the eighth bit specifies whether is was a key press or a key release.

**7.** What is the difference between raw and cooked modes? **ANS:** Many applications do not really want to see every character typed. For example, you shell would rather wait until a full line is assembled, and then receive an entired command one enter is pressed. Thus, you can type part of a line, erase some characters, type some more, and the shell will never know of your indecisiveness; it will only see the corrected line, after the entire line is assembled. This is sometimes called ‘cooked’ mode. Raw mode does not wait for entire lines, but sends each character to the process as they occur.

**8**. What use is /etc/termcap? **ANS:** The appropriate codes for clearing the screen, deleting a character, moving the insertion point, opening up a line in the middle of the screen, doing a non-destructive backspace, and so on are stored in a huge database with different command sequences for different styles of terminals. This is the “terminal capabilities” database, typically stored in /etc/termcap.

**9.** Name and define the three main components that make up the X window GUI. **ANS**: Server -- The ‘server’ is responsible for driving the hardware, it decides what actually gets’ drawn’ on the screen. Applications -- various applications, such as ‘xbiff’, create windows. There is a temptation to think of these utilities as servers, since they ‘serve up’ information from various remote locations. However they are essentially ‘clients’ of the server that drives the display hardware. Window managers -- The window manager is responsible for window’ decorations’ that allow the windows to be moved, resized, iconified, etc.

**10**. What is the defining difference between block- and character-oriented devices? **ANS**: Devices come in many flavors, but most of them can be lumped into two general categories. Once category is block-oriented, which handles data in fixed-size ‘chunks’, and the chunks can be accessed(read or written) independently. These types of devices called ‘block devices’ include things like disk drives. The other main category consists of ‘character devices’ which input or output a stream of data. For these, you cannot seek to an individual position.(for example a mouse is a character device)

E(ch10) **1.** how is dynamic linking different from static linking? what are the advantages of using dynamic linking? **ANS:** Static linking takes all library modules used in a program and loads them into the final executable. Dynamically linked libraries just take the name of the external modules and links the associated files at runtime. Dynamic linking files probably load faster, since what you need is already in memory, and better use of RAM as well, thus reducing the size of the executable and saving memory and disk space. It’s also more secure because the linked libraries can receive system updates instead of being baked into the executable.

**2.** Name a system call that will cause the [non-writable] text area of a process to suddenly be different. Hint: your shell made extensive use of this fundamental magic**. ANS:** Execvp() When you execvp() all the code from the parent is dumped and replaced with whatever program is execed. The text area is obviously changed because it must contain the executable for the new process.

**3.** What [and how] is uninitialized data treated differently from initialized data in the a.out file? **ANS:** Uninitialized is not actually copied from the disk because by default uninitialized data is all null. There is no point in storing a bunch of nulls. Therefore, the linker just allocates the amount of data which will ultimately be needed for these uninitialized variable but does not actually copy these nulls from the disk. Initialized data is loaded into memory with the executable, and stored in the data section.

**5.** Why do text/data/stack segments have different writable/executable permissions? **ANS:** This is for protection. One would not want the stack to have executable permissions because a hacker could use their own input, which can be machine code, to redirect the return address of a system function to instead jump to some other instructions. The text segment is the opposite, it can execute but not write. This is because once it is executing, you do not want the code in this to be modified.

**6.** How does the Linux Page Frame Reclaiming Algorithm (PFRA) differ from our original Clock algorithm? **ANS:** The PFRA first distinguishes pages by 3 categories: discardable, syncable, and swappable. It first checks if any pages are discardable and puts them on the free list immediately. After that, with the two remaining “harder” categories(syncable and swappable), the PFRA then using a clock algorithm to figure out which pages among them to free up.

**7.** Describe the UNIX virtual file system, and explain why it is useful. **ANS**: The purpose of the VFS is to allow client applications to access different concrete file systems in a uniform way. For example VFS can be used to access network and local storage devices transparently without client application noticing the difference. Or it can be used to bridge differences between Windows, Unix, and Mac OS file systems.

**DMA(Direct Memory Access)** The DMA controller is programmed with a physical address within a page frame, and then the CPU goes off and attends to other matters, trusting the DMA to carry out the transfer once the data is assembled in its buffer.

**What is the benefit of DMA?** The CPU can do other tasks while the DMA controller handles a slow task like reading from disk.

**How was disk access handled before DMA?** The CPU oversaw the entire disk transfer byte by byte.

**What special considerations need to be made for Flash Memory?** Wear-levelling- flash memory bits will stop functioning after a certain age, so data writing and reading has to be spread across the disk to extend the lifespan of the hardware.

**GPT** stands for GUID Partition Table. ... GPT allows for a nearly unlimited amount of partitions, and the limit here will be your operating system — Windows allows up to 128 partitions on a GPT drive, and you don't have to create an extended partition. On an MBR disk, the partitioning and boot data is stored in one place.