File Systems

1. What is a File system? Describe its three primary responsibilities.

2. What's an i-node? Where is it stored?

3. What's the simplest data structure for an i-node? Then why is UNIX i-node so complicated?

4. In a file-system, (a) What is meta-data? (b) Where is meta-data stored? (c) Why is it important for a file system to maintain the meta-data information? (d) List some of the typical information that is part of the meta-data.

5. If you collect a trace of I/O operations below the file system cache (at device driver or physical disk level), what type of I/O operations do you expect to see more of -- write I/O requests or read I/O requests? Explain why.

6. (a) Suppose you collect a trace of I/O operations above the file system layer (in applications or in system calls). Do you expect to see more write I/O operations or read I/O operations? (b) Now suppose you collect a similar trace of I/O operations below the block device layer (in the disk or device driver). Do you expect to see more write I/O operations or read I/O operations? Explain why?
7. If you increase or decrease the disk block size in a file system, how (and why) will it affect ***(a)*** the size of the inode, and ***(b)*** the maximum size of a file accessible only through direct block addresses?
8. How does the inode structure in UNIX-based file-systems (such as Unix V7) support fast access to small files and at the same time support large file sizes.
9. What does the file system cache do and how does it work? Explain with focus on the data structures used by the file system cache.
10. Explain the role of *file system cache* during (a) read I/O operations and (b) write I/O operations.

11. Describe two different data structures using which file system can track free space on the storage device. Explain relative advantages/disadvantages of each.
12. How does a log-structured file system work? How is its performance dependent upon the file system cache (or page cache) maintained by the OS?

13. In a file-system, explain how two different directories can contain a common (shared) file. In other words, how do hard links work?
14. How does the inode structure in UNIX-based file-systems (such as Unix V7) support ***fast access to small files*** and at the same time ***support large file sizes***.
15. Explain the structure of a UNIX i-node. Why is it better than having just a single array that maps logical block addresses in a file to physical block addresses on disk?
16. Explain the steps involved in converting a path-name */usr/bin/ls* to its i-node number for the file *ls*.

17. What's wrong with storing file metadata as content within each directory "file"? In other words, why do we need a separate i-node to store metadata for each file?

18. Assume that the
- Size of each disk block is B bytes.
- Address of each disk block is A bytes long.
- The top level of a UNIX i-node contains D direct block addresses, one single-indirect block address, one double-indirect block address, and one triple-indirect block address.
  (a) What is the size of the ***largest "small"*** file that can be addressed through direct block addresses?
  (b) What is the size of the ***largest*** file that can be supported by a UNIX inode?

Explain your answers.

19. In a UNIX-like i-node, suppose you need to store a file of size 32 Terabytes (32 * 240 bytes). Approximately how large is the i-node (in bytes)? Assume 8096 bytes (8KB) block size, 8 bytes for each block pointer (entry in the inode)., and that i-node can have more than three levels of indirection. For simplicity, you can ignore any space occupied by file attributes (owner, permissions etc) and also focus on the dominant contributors to the i-node size.
20. In a UNIX-based filesystems, approximately how big (in bytes) will be an inode for a 200 Terabyte (200 * 240 bytes) file? Assume 4096 bytes block size and 8 bytes for each entry in the inode that references one data block. For simplicity, you can ignore intermediate levels of indirections in the inode data structure and any space occupied by other file attributes (permissions etc).
21. In a UNIX-based filesystems, approximately how big (in bytes) will be ***an inode*** for a ***400 Terabyte (400 * 2^{40} bytes) file***? Assume 4096 bytes (4KB) block size and 8 bytes for each entry in the inode that references one data block. For simplicity, you can ignore intermediate levels of indirections in the inode data structure and any space occupied by other file attributes (owner, permissions etc).
22. Assume that the size of each disk block is 4KB. Address of each block is 4 bytes long. What is the size of the ***largest*** file that can be supported by a UNIX inode? What is the size of the ***largest "small"*** file that can be addressed through direct block addresses? Explain how you derived your answer.

23. Assume all disk blocks are of size 8KB. Top level of a UNIX inode is also stored in a disk block of size 8KB. All file attributes, except data block locations, take up 256 bytes of the top-level of inode. Each direct block address takes up 8 bytes of space and gives the address of a disk block of size 8KB. Last three entries of the first level of the inode point to single, double, and triple indirect blocks respectively. Calculate **(a)** the largest size of a file that can be accessed through the direct block entries of the inode. **(b)** The largest size of a file that can be accessed using the entire inode.

24. In the "UNIX/Ritchie" paper, consider three major system components: files, I/O devices, and memory. UNIX treats I/O devices as special files in its file system. What other mappings are possible among the above three components? (In other words, which component can be

treated as another component)? What would be the use for each possible new mapping?

25. Suppose your filesystem needs to store lots of uncompressed files that are very large (multiple terabytes) in size. (a) Describe any alternative design to the traditional UNIX inode structure to reduce the size of inodes wherever possible (NOT reduce the file content, but reduce inode size)? (Hint: maybe you can exploit the nature of data stored in the file, but there may be other ways too). (b) What could be the advantage of your approach compared to just compressing the contents of each file?
26. Why doesn't the UNIX file-system allow hard links (a) to directories, and (b) across mounted file systems?

27. Why did the authors of the "UNIX" paper consider the UNIX file-system to be their most important innovation?

28. Consider a UNIX i-node for a file of size F bytes. What is the size of the i-node in bytes? Assume that disk block size is B bytes, each block address size is A bytes. The top level of the i-node contains D direct block addresses, one single-indirect block address, one double-indirect block address, and one triple-indirect block address.