Preparing for the Exam: Questions and Answers

1. When does the system call read() return 0?
Answer:
A value of zero indicates end-of-file (except if the value of the *size* argument is also zero). This is not considered an error. If you keep calling read while at end-of-file, it will keep returning zero and doing nothing else.
If read returns at least one character, there is no way you can tell whether end-of-file was reached. But if you did reach the end, the next read will return zero.

2. Is the following line of code is legit? `lseek(fd, (long)5, SEEK_END);`
Answer:
Yes. You can set the file position past the current end of the file. This does not by itself make the file longer; lseek never changes the file. But subsequent output at that position will extend the file. Characters between the previous end of file and the new position are filled with zeros. Extending the file in this way can create a "hole": the blocks of zeros are not actually allocated on disk, so the file takes up less space than it appears so; it is then called a "sparse file".

3. What is an *inode*?
Answer:
An inode is a data-structure used by a file system to store important information about a physical file on the hard disk. (need to say 'node' or 'data structure; inode is **not** a 'number')

4. How can you test whether a pathname is *absolute* or *relative*?
Answer:
If the first character is / (not '\'!), then the name is an absolute path. All others are relative.

5. Is a directory file a regular file?
Answer: No.

6. If you have execute-permission for a directory, can you delete a file in that directory (Yes/No)?
Answer:
No. You need write permission on the directory. Write permission on file **not** necessary.

7. Consider the . and .. special files. Is it correct to say that these files are system-created hard links to directories?
Answer: Yes.

8. Joe wants to print out a numbered listing of the file "fa". He types "`cat -n fa > lpr`" but no printout appears. Why? What command would you suggest he use?
Answer:
He has redirected stdout into a file named "lpr"
He should have written: `cat -n fa | lpr`

9. Briefly explain the I/O redirection occurring in each of the following commands:
`cmd1 | cmd2`
`cmd1 |& cmd2`
Answer:
The first connects stdout from cmd1 to stdin for cmd2. The second connects both stdout and stderr from cmd1 to stdin for cmd2.

10. Why is a hard link indistinguishable from the original file itself?
Answer:
The "original file" is really just a hard link itself to the file's inode. If you add a new hard link, it will behave identically.

11. What happens if you rm a hard link?
Answer:
The hard link is removed, and if the file's (inode's) hard link count goes to 0, the file is also removed.

12. Why is it not possible to have a hard link to a file in a different file system?
Answer:
A hard link need to specify the inode of the linked file, and it is assumed that the inode is in the same file system (there is no mechanism for specifying an inode and a file system in the link).

13. What is the difference between an absolute and relative pathname? Give an example of each.
Answer:
Absolute pathnames start with '/' and specifies a path starting from the root directory *(e.g.* /bin/csh). Relative pathnames specify a path starting in the current working directory *(e.g.* ../temp/myStuff.txt)

14. What is the command to create a soft link "slink" to the file "/u/maclean/junk.txt"?
Answer:
```
ln -s /u/maclean/junk.txt slink
```

15. What is the value of argc and argv[2] in the C program invoked as
"`myProg -13 fred < csc209.lst > csc209.marks`"?
Answer:
argc = 3
argv[2] = "fred"
(Pay attention to argc value… it's important to know that the program never "sees" I/O redirection …)

16. Briefly explain the I/O redirection occurring in each of the following commands:

```
myCmd > someFile
```
Answer:
output (stdout) from "myCmd" goes into file "someFile" (created if necessary, or overwritten if it exists & "noclobber" is not set)

```
myCmd < someFile
```
Answer:
input (stdin) for "myCmd" comes from file "someFile"

```
myCmd >> someFile
```
Answer:
output (stdout) from "myCmd" is appended to file "someFile"

```
myCmd >! someFile
```
Answer:
output (stdout) from "myCmd" overwrites "someFile", even if "noclobber" is set

```
myCmd >& someFile
```
Answer:
output (stdout + stderr) from "myCmd" is written to "someFile"

17. What is the relationship of an inode to the name for a file? Is it possible for the same inode to be referred to by different filenames? Explain.
Answer:
An inode is the file. The filename is merely an identifier in a directory which references an indoe. Yes, it is possible for the same inode to be referred to by different file names. A Unix hard link accomplishes this.

18. It has been suggested that the first part of each Unix file be kept in the same disk block as its inode. What benefit - if any - would this provide?
Answer:
Many files in a Unix system are quite small - smaller than the size of a disk block. Accessing a file in Unix requires (1) accessing the disk block containing the inode (2) reading the first (potentially of many) disk block. If the first part of each file were kept in the same disk block as its inode, this would mean that for small files you would only have to go to disk once to access the file.

19. A hard link cannot span different file systems in Unix. Why not? (Hint: Think of the relationship of inodes to different file systems.)

Answer:

Each file system in Unix has its own inode list. Therefore, if a particular computer has 3 different file systems, there may be 3 different inodes with the number 172 (for example.) A hard link creates another link to an inode. If hard links were allowed to span file systems and you created a hard link to inode 172, which file system would you be referring to?

20. What is the *super block*?

Answer:

A file system in Unix contains:

Boot Block	***Super Block***	***Inode List***	***Disk Blocks***

The super block describes the state of the file system - how large it is, how many files it can store, where to find free space on the file system, etc. In particular it contains:

- size of the file system
- number of free blocks in the file system
- list of free blocks in the file system
- index of the next free block in the free block list
- the size of the inode list
- number of free inodes in the file system
- list of free inodes in the file system
- index of the next free indoe in the free inode list
- lock fields for the free block and free inode list (to prevent race condidtions)
- a flag indicating that the super block has been modified.

21. Explain why the Unix rm command calls the unlink command. When is a file actually removed (i.e. the inode and disk blocks returned the the free list?)

Answer:

One of the fields in an inode is the number of links to the inode. When the rm command is performed on a file, it only decrements the link count (by calling the unlink command.) Only when the link count goes to zero are the inode and disk blocks returned to their respective free lists.

22. A variation of an inode is called a *linode* (short for little inode.) Assume the size of a disk block is 4K and a disk block may hold 2K disk block addresses. A linode has the following structure:
entries 0 - 6 are direct disk block pointers.
entry 7 is a single indirect.
entry 8 is a double indirect.
entry 9 is a triple indirect.
What is the maximum size of a file (in bytes) using a linode?

Answer:

It would be (7 * 4K) + (2K * 4K) + (2K * 2K * 4K) + (2K * 2K * 2K * 4K)

23. A well known computer scientist is fond of saying 'to get good performance out of disks you have to realize they are really sequential not random access devices' What does he mean by this statement?

Answer:

Disks have heads that need to seek to search the desired blocks. Thus the time to access random locations is not uniform and independent of the previous accesses.

24. If you knew that most accesses to the file system were either backward sequential reads, or append writes to very large files (several gigabytes each) which file-system implementation would you prefer: linked allocation (MS-DOS) or indexed allocation (UNIX)? Explain your answer.

Answer:

Indexed allocation is better since it is possible to reach any location in a file with only a constant number of pointer accesses (through the direct, indirect, double indirect or triple indirect blocks). With linked allocation we need to walk the forward through the links to find a specific location in a file. This applies to reverse sequential reads. For appends, most linked schemes maintain a pointer to the last block to make append faster.

25. A user has write privilege to a UNIX directory "goodStuff", but not to the file "protected" in the "goodStuff" directory. Explain how the user can still modify the file "protected".

Answer:

cp goodStuff temp
rm -f goodStuff
mv temp goodStuff

26. A UNIX system has just been rebooted. What is the minimum number of disk blocks that must be read to get to the millionth byte of the file "/a/big/file", assuming the file-system uses 4KB blocks, and 2-bit indoe numbers?

Answer:

1. read root inode (#2)
2. read root directory and find "a"
3. read inode for "/a"
4. read directory chunk and find "big"
5. read inode for "/a/big"
6. read directory chunk and find "file"
7. read inode for "/a/big/file"
8. read indirect block for the file (since 4KB blocks hold 1,024 inodes, files up to 4MB are in direct or indirect block)
9. read the data block

27. Explain why a bit vector implementation of a free block list can provide increased reliability and performance compared with keeping a list of free blocks where the first few bytes of each free block provide the logical sector of the next free block.

Answer:

Performance: bit vectors provide fast access to find clusters of adjacent free blocks.
Reliability: if an item in a linked list is lost, the rest of the list is lost. With a bit vectors only the items are lost. Also, it's possible to have multiple copies of the bit vector since it is a more compact representation.

28. ברצונך ליצור קובץ במערכת UNIX כך שרק יוסי יכול לקרוא אותו, רק יעל תוכל לכתוב אותו, ובנוסף שלך תהיה אפשרות גם לקרוא וגם לכתוב. הפתרון הוא:

א. הרשאת rw לך בתור בעל הקובץ, r לקבוצה המכילה את יוסי, ו-w לקבוצה המכילה את יעל.
ב. הרשאת r לקבוצה המכילה אותך ואת יוסי, והרשאת w לקבוצה המכילה אותך ואת יעל.
ג. הרשאת rw לך בתור בעל הקובץ, w לקבוצה המכילה את יעל ו-r לשאר המשתמשים.
ד. אין אפשרות ליצור הרשאות כאלה.

תשובה: ד'.