Salgado Berardo, Joao Antonio 100280950 A4 Jun/13/2017 Mark:_____

Chapter 3: 3.1. 3.2, 3.3, 3.5, 3.6, 3.7, 3.8, 3.9, 3.12 and 3.15

Chapter 3:

3.1 The directory file contains the index node (inode) and name of files and subdirectories under it.

When you use the cp command, if the destination file doesn't exist, it will be created and will have its new inode number and name in the directory file. The source file remains unchanged.

With the mv command, the source file has its inode and name removed from the directory file and the details of the destination file will be included in the directory file.

Using the rm command removes the details (inode and name) of files being removed.

The directory size is small because it actually does not contain the files itself, it contains the details related to files. It contains the inode and the file name and the inode has other details related to the file.

3.2 Device files are located under /dev directory. We have two types: character files and block files.

If a user connect an external hard disk, a device file is created for this hard disk inside /dev directory.

This way, users can read and write files in the external hard disk without worrying with the low level implementation of reading and writing files to/from hard disk.

3.3 mkdir a/b/c: Does not work.

mkdir a a/b: It works because it first creates the directory ‘a’ and then it creates the directory ‘b’. If the command was ‘mkdir a/b’ it would not work because the directory ‘a’ does not exist. To create both at the same time, the command should have the option -p.

rmdir a/b/c: This command works and it removes the directory ‘c’ under a/b. One thing to note is that the directory must be empty before deleting it. One can use -f option to force removing even if it is not empty.

rmdir a a/b: Does not work on a, but works on b. The rmdir command tries to remove the directory ‘a’ first, but because it is not empty, it fails. The second argument (a/b) works and removes the directory ‘b’. If the order was changed, and directory ‘b’ was the only content inside directory ‘a’, the command would work (rmdir /ab a).

mkdir /bin/foo: Does not work.

3.5 ‘echo “test” > .’ Doesn’t work

‘echo “test” > ..’ Doesn’t work

‘echo “test” > ...’ Works

Only the three dots file worked in that case. This is because one dot and two dot files are actually directories. They exist in every UNIX directory. One dot represents the current directory and two dots the directory above it (parent directory).

3.6 To see all contents inside a directory, it is best to use ls -la. In this case, it shows dot files that are hidden using only ls. Probably the command failed because the directory bar contains at least one dot file.

3.7 (i) If bar2 exists and the user has privileges, the command will move the directory bar1 to bar2

(ii) If bar2 doesn't exist, the mv command will rename bar1 to bar2

3.8 cd ~charlie changes to the charlie's home directory.

cd ~/charlie tries to change to the directory named 'charlie' inside the charlie's home directory (for example: /user/charlie/charlie).

It is possible for both commands to work, assuming all directories exist.

3.9 He could have used the environment variable HOME in his scripts.

3.12 The command 'ls -d' lists the directory itself. From the man page:

"-d, --directory: list directories themselves, not their contents"

The two ways I can think of are:

1. The user is in a directory, for example foo and the directory bar is inside foo. For the output to be only the string bar using ls, the user should be under foo and the foo directory must contain only the bar directory. In that case, ls output will be bar and ls -d output will also be bar.

2. The only other way I could think of is issuing the ls and ls -d command using absolute pathnames and foo must contain only bar directory.

3.15 The user don't have permission on the folder.

Even if the files exist, the user could be in the wrong directory, the command will fail.

The destination file has only permission to read the file.