6.033 Hands-on Unix

Kevin Cho
R5 Peter Szolovits 1PM

Due February 16th 2016

I Pipe Questions

1. *ps -o comm –no-headings – sort=cmd -u kcho13*
2. *grep -c -v '[aeiou]' /usr/share/dict/words*
3. *yes "1 0" — fmt -w 10 — head-6*
4. *ls -a -l -S /etc/*.conf* — tail -5*
5. For the second command given, we get the error of having the permission denied to access the files. This is because we need the access to write in the /etc folder when we actually don't have the access. Also, the second command can cause a different output from the first sicne we are adding a new file to the directory called temp. If the head commmand was used for a larger integer: i.e *head -20*, then we could possibly have temp inside the head for the second command while the first command gives us the first 20 files and directories without creating temp. A negative side effect for the second command is that the temp file coudl be overwritten if there was already one in side the working directory. This would cause a lose of information that was stored in the temp file.
6. The difference in the two temp files are cause by concurrency. When we use the semicolon ";", we are actually running the first program and then running the second one after the first is over. Thus, we get the file of "ynyn" because the first one prints out "yn" and the second does the same. However, for the and sign "&", we are running the two files in parallel. This is why the temp file for this command is "yynn" because the first and second are running together and both print out "y" at the same time and "n" at the same time.
7. The offset was not included because the file systems were designed to be stored as a number of files linked in a memory block. Thus, this design makes it hard to seek out the specific offset without reading through each file, one memory block at a time. This is why the designers thought it wouldn't be a great idea to include the offset as an argument. An application would write to a specific offest by seeking and setting the cursor to the offest and write on the file what we want the application to write.

II Creating Directories and Files

8. For the '.' entry, nothing happens to the working directory. We stay where we are currently at. For the '..' entry, we move back up one parent directory. Thus, our working directory changes to the parent directory.
9. A scenario would be when you need to call the current working directory in a command. One such command would be copying files or directories from another pathway to the working directory. Then, we would use [*cp (file/directory to copy) .*].
10. The timestamps on the access, modify, and change sections were changed. These changes were caused because by creating the foo and bar files, we have accessed, modified, and changed the current working directory.

11. The number of links changed along with the timestamps of access, modify, and change.The number of links changed because by creating a new directory, we have created a new pathway from the current directory to a different. Thus, we have obtained another ”link” to a new path. The timestamps have changed due to the same reason as #10. The link count only changes when we create a new directory because for files, we don’t create a new pathway. We only create a new pathway when we create directories, and thus, we create a link.

III Creating Links

12. One advantage of symbolic links is that we would be able to access the directory from anywhere we would like. You can create symbolic links to link directories from anywhere in the system. One disadvantage is that if we move the directory, the symbolic link would not know about this change.

13. We get the error that no such file or directory exists. This is because when we followed the symbolic link, we ended up in a totally different pathway (/afs/athena.mit.edu/course/6/6.033). Thus, when we do the cd../kcho13 command, we look for the directory of kcho13 in /afs/athena.mit.edu/course/6, which no such directory exists.

14. It provides an easier way to access the 6.033 locker instead of cd’ing to the actual location.

15. We notice that the ’..’ entry provides a pathway to the immediate parent directory. However, if we can change the filesystem such that the ’../(anything)’ entry for cd searches not only the immediate parent directory but all relevent directories such as symbolic links, we would not have the problem we have right now. However, a change in the filesystem like this can cause some confusion such as if the (anything) directory is actually a directory in several valid places, which one would we choose? These are some tradeoffs that would be made if we changed the filesystem.

IV The Search Path

16. When we overwrite $PATH to have the period in the beginning, we are essentially calling the bash calls on all files. Thus, when we do the ’ls’ command, we are actually not calling the standard ’ls’ of getting the items in the working directory. We are running the ls bash script inside our working directory which causes us to get the output we got.

V System Calls

17. The 1 represents the filep arguement, which is the file descripter. The file descripter is the number used to identify which file the buffer should be written into.

18. The assignment took about 2 and a half hours to finish.