6.033
COMPUTER SYSTEMS ENGINEERING
HANDS ON 2: UNIX

JOHN WANG

1. Pipe Questions

(1) *ps -o comm -u wangjohn –no-headers — sort*
(2) *echo "(cat/usr/share/dict/words|wc − l)−(cat /usr/share/dict/words — grep -c '[aeiou]')" — bc.* This command first looks for all the lines in dictionary, then subtracts the number that match the regular expression '[aeiou]', and subtracts the first output from the second output by piping the command to a built in calculator.
(3) *yes "1 0" — fmt -10 — head -5.*
(4) *ls -l -a -S -d /etc/*.conf* /etc/.conf* — head -5*
(5) The second command does not work because it needs write access to the /etc directory in order to create the file named temp. However, the first command pipes the ouput from ls directly to the head command without the use of a temporary file. The output could also differ in these two commands because the second command creates the temp file, which means temp could appear in the second command while not appearing in the first. Some negative side affects of the second command are that one might overwrite a file already named temp if it is in the folder, and also that the temporary file is left behind.
(6) There is a concurrency issue here. In the first temp file, the process waits for the first myyes to finish before starting the second myyes. However, in the second temp file, both myyes files start executing at the same time. Therefore, the first file has the "y n" pattern repeated, while the second has two "y"s then two "n"s.
(7) The UNIX designers probably did not include an offset as an argument to read and write because the filesystem is stored as a number of files linked together inside of a block. Therefore, it would be extremely difficult to seek a particular offset without reading through the file one memory block at a time. Thus, it doesn't provide any extra functionality to have the offset as an argument. To write to a specific offset, one would first have to seek and set the cursor to a specific offset, then write to the file.

2. Looking Around

(8) Changing to the '.' entry does not change the working directory, but changing to the '..' entry moves the working directory up to the parent of the previous working directory.
(9) One might want to copy a directory into the current working directory like so *cp sharedfolder .* and the '.' provides this ability.
(10) The access, modify, and change timestamps on *stat .* have changed because the directory has been modified, accessed, and changed by creating the two new files "foo" and "bar."
(11) The link count as well as the access, modify, and change timestamps have changed. The timestamps have updated while the link count has incremented by 1. The link count only changes when a new directory is created because the new directory creates a link between its '..' file and the original directory's '.' file. This allows the new directory to be navigated and also increments the original directory's link count.

3. Creating Links

(12) One disadvantage of symbolic links on the same disk is that if you move the original file, then the symbolic link will not know that this file has been moved (since symbolic links only use the filename). One advantage of using symbolic links is that they can refer to directories. Thus, one can create a shortcut to a directory that is very far down the directory tree structure.

(13) The command sent me to my home directory. This happened because the *cd* command sent me up one directory to /mit, then changed the directory to /mit/wangjohn, which allowed me to get to my home directory.
(14) Athena provides the /mit/6.033 symbolic link to make it easier for a user to navigate around the filesystem. Typing out the full absolute path is a pain, and /mit/6.033 is much faster and easier to remember.
(15) One could change the filesystem so that /mit was a directory in the home folder of athena. This would mean that using .. on /mit/6.033 would send you up into the /mit folder, from which you can then descend into /mit/wangjohn.

4. The Search Path

(16) Inside of the working directory of /mit/6.033, there was a bash script named "ls" which overwrote the built-in *ls* command. This is because the first path inside of the PATH variable was the working directory, so it ended up running the bash scrpt which was defined inside of the current working directory.

5. System Calls

(17) The 1 in the first argument of write is the file descriptor to which the contents of the buffer will be written to.
(18) The assignment took me 3 hours.