Assignment 3 Answers

Week 3 Assignment - note that the output might be a little different from system to system. There might be some extra questions from previous assignments that will worth while looking at them.

==
Question 1
==
Give three different commands to get back to your home directory.

---------------------------------- Answer -----------------------------------
We accomplish getting back to the home directory by using the cd command
with one of many different options passed in such as the following:

```
> cd
> pwd
/home/student/john.doe
> cd ~
> pwd
/home/student/john.doe
> cd ~john.doe
> pwd
/home/student/john.doe
> cd $HOME
> pwd
/home/student/john.doe
>
```

==
Question 2
==
Show the names of all hidden files in your home directory. Show your session.

---------------------------------- Answer -----------------------------------
To get the listing of hidden files, we specify that we want "All" files from
the output of the ls command, and then from those results, only those that
match a particular pattern. Specifically, those that start with a period (.)
at the beginning of the line. Example:

```
> ls -a | grep '^\.'
.addressbook
.bash_history
.bash_profile
.bashrc
```

```
.emacs
.gnu-emacs
.inputrc
.lesshst
.pinerc
.profile
.viminfo
.vimrc
.xim.template
```

==
Question 3
==
Write down the following and show the session that you used to obtain the required information:
Your user name (also known as login name)
Your user ID
Your group ID and group name
User names of users in your group
Number of groups that you are a member of

---------------------------------- Answer -----------------------------------
To get the information needed we will use a series of commands. The id command will be used to find out information about ourselves, and then we will interrogate two linux system files /etc/passwd and /etc/groups to get information about who is in our groups. Note, we have to use BOTH files because the correct resulting set should be the UNION between both entries.

The reason is because entries found in the /etc/groups file will be for those users who are in the group, but NOT as their default group, and similarly, the information in the /etc/passwd will only contain information about the users default group.

To get this output nicely, we combine commands on the command line via the shell's grouping methodology which in this case joins the output from two grep results, and finally sorts it nicely for us. For each of the grep commands, the use of the cut and tr commands are used to "pull out" just the values (usernames) that we are interested in.

```
> id
uid=3182(john.doe) gid=100(users) groups=100(users)
> id -Gn | wc -w
1
> (grep ^users /etc/group | cut -f4- -d':' | cut -f1- -d',' | tr ',' '\n' ; grep 100: /etc/passwd | cut -f 1 -d':') |
sort
aaron.brown14
aaron.fifer
aaron.herinckx
aaron.leondar
aaron.miller5
aaron.squier
abdallah.nawras
abdel.ouedraogo
abdi.sharif1

--- cut ---

zackery.salzwedel
zakk.fields
zara.elmahjour
zena.martin
zhengshan.low
zhong.zheng
zhongjie.huang
>
```

This results in the following values for myself, John Doe:

Username (Taken from the first cmd output)
john.doe
UserID (Taken from the first cmd output)
3182
GroupID (Taken from the first cmd output)
100 (users)
Usernames in Users (Taken from the third cmd output)
aaron.brown14
--- 1,651 entries later ---
zhongjie.huang
Number of groups which a member of (Taken from the second cmd output)
1

==
Question 4
==
What are the permissions for your home directory set by your system administrator (not the directory called /home, not the permissions of the individual files in your home directory)? What command did you use to answer the question? Show your
session.

---------------------------------- Answer -----------------------------------
To get the permissions of my home directory, we first found out what our home directory is on the file system (which we did in question 1), by using a "longer" version of the ls command:

```
> ls -ld /home/student/john.doe
drwx------ 13 john.doe users 4096 Apr 18 23:07 /home/student/john.doe
```

As a result, the permissions are:
For the User: Read, Write, and Execute (full permission)

For the Group(users) None
For Everyone(all) None

==

Question 5

==

Suppose that you want to block everyone (except you) at the main door (your ~ directory) of your directory hierarchy (i.e., your home directory-not the files inside the directory), as shown in Figure Lab7.1 (below). Take the necessary steps to do this. Show the session that you used to accomplish the task and confirm that the task has actually been done.

---------------------------------- Answer -----------------------------------

To accomplish a full lock down of our home directory (and below), we use the chmod command to set/restrict the settings for just the user. Question 4 already shows our directory is in this restricted mode, but to illustrated, I will change (open) the directory up for all members of the users group to view the files first, then lock it down, and show the settings after each:

```
> chmod 0750 /home/student/john.doe
> ls -ld /home/student/john.doe
drwxr-x--- 13 john.doe users 4096 Apr 18 23:07 /home/student/john.doe
> chmod 0700 /home/student/john.doe
> ls -ld /home/student/john.doe
drwx------ 13 john.doe users 4096 Apr 18 23:07 /home/student/john.doe
```

==

Question 6

==

What do . (dot), .. (dotdot), and ~ represent in UNIX? Show use of all three.

---------------------------------- Answer -----------------------------------

the . (dot) directory represents the current directory we are in when used in specifying a file path. The .. (dotdot) command represents the directory level ONE higher or above than our current directory, and lastly the ~ (tilde) represents our home directory. We can see this by starting in our home directory and traversing up, one level at a time, stopping to display the current directory we are in:

```
> cd ~
> pwd
/home/student/john.doe
> cd .
> pwd
/home/student/john.doe
> cd ..
> pwd
/home/student
```

==

Question 7

==

What is an inode number in UNIX? Show the command for displaying the inode numbers for the root directory and your home directory (not the directory called /home).

---------------------------------- Answer -----------------------------------

the iNode is a number in UNIX that is used to represent a data structure that contains information about a file such as the file's length, date/time of when it was last accessed and modified, owner and group ID, access privileges, number of links (references to that file), and pointers to the actual data blocks on the disk for that file's contents.

Each directory entry entry associates a filename with an iNode. Although a specific file on disk may have many filenames associated with it (via hard and sym links), the file itself will only ever have ONE iNode.

To view some sample values of the iNode entries for contents of the root and our home directory, we can use the -i option of the ls command:

```
> ls -i /
388611 bin 129537 etc 388634 lib64 389045 mnt 388755 root 389220 selinux 389223 tmp
2 boot 2 home 11 lost+found 389046 opt 1174 run 388850 srv 119 usr
1025 dev 12 lib 389044 media 1 proc 388757 sbin 1 sys 259075 var
> ls -ldi ~john.doe
191134 -rw-r--r-- 1 john.doe student 16 Oct 31 2013 john.doe
```

==

Question 8

==

Give the command for removing an empty directory called personal under your home directory. How would you remove it if the personal directory is not empty?

---------------------------------- Answer -----------------------------------

To remove an empty directory, such as one called personal, we would use the rmdir command as in the following example:

```
> cd
> mkdir personal
> ls -lrta personal/
total 8
drwx------ 14 john.doe users 4096 Apr 24 20:29 ..
drwx------ 2 john.doe users 4096 Apr 24 20:29 .
> rmdir personal
> ls personal
ls: cannot access per: No such file or directory
```

However, this command would fail if there were contents, files or worse, other directories below it. If this were the case, we would have to remove files one at a time, recursively. To do this, we would use the rm command with the recursive option (-r).

```
> cd
> mkdir personal
> echo "Hello World" > ~/personal/test.txt
> rmdir personal
rmdir: failed to remove 'personal': Directory not empty
> rm -r personal
> ls personal
ls: cannot access personal: No such file or directory
```

===
Question 9
===
Create a file, called sample, in your home directory and set its permissions to read and write for yourself, read for users in your group, and none to everyone else. What command did you use to create the file? What command did you use to set privileges?

---------------------------------- Answer -----------------------------------
To simply create a file, or as we seen from question 7 an inode within the file system, we can simply use the touch command and specify the file we want created. Once created, we use the chmod command to alter the security settings on the file in order to allow the owner (ourselves) full control, and at the same time, allow others members of the users group the ability to read the file. An example of the commands in use is:

```
> cd
> touch sample
> ls -lrt sample
-rw------- 1 john.doe users 0 Apr 24 20:37 sample
> chmod 740 sample
> ls -lrt sample
-rwxr----- 1 john.doe users 0 Apr 24 20:37 sample
```

===
Question 10
===
Suppose that umask is set to 022 on a system. What will be the default permissions for the new directories and text files that you create? Explain your answer.

---------------------------------- Answer -----------------------------------
If the umask is set to 022 on a system, then the default permissions for newly created inodes (files or directories) will be 755, which would mean that the user/owner of the inode would have full permissions (read, write and execute), while members of both the group for which the iNode belongs to would have the only the read and execute privileges (no write). The same would hold true for all users, or the everyone group.

How this is determined is by using the octal representation of each of the permissions that are established, and subtracting the mask value from it. For example, when looking at the directory entry from question 9 above, we find its permissions value to be:

```
> ls -lrt sample
-rwxr----- 1 john.doe users 0 Apr 24 20:37 sample
```

forgetting the first dash (-) character, which specifies if the entry is a directory or not, the very next clust of 3 letters represents the users permissions such that:

r = 4 (binary value for a 1 in the 2nd position, with the first being 0)
w = 2 (binary value for a 1 in 1st position)
x = 1 (binary value for a 1 in the 0th position)
=====
Sum 7

The same thing would apply to the next 3 values in the listing, which corresponds to the group settings:

r = 4 (binary value for a 1 in the 2nd position, with the first being 0)
w = 0 (binary value for a 0 in 1st position)
x = 0 (binary value for a 0 in the 0th position)
=====
Sum 4

And lastly the final cluster of 3 values is for the everyone or all group, and using the same values, amounts to 0, or no permissions whatsoever.

When iNodes (files or directories) are created, the permissions are assumed to be wide open for everyone, with a value of 777, which again breaks down like this:

r = 4 (binary value for a 1 in the 2nd position, with the first being 0)
w = 2 (binary value for a 0 in 1st position)
x = 1 (binary value for a 0 in the 0th position)
=====
Sum 7

However, this is only the initial assumption and the system then SUBTRACTS what ever the umask value is for the owner who is create the file using the umask value as the amount to subtract for each cluster of three (user, group, all).

So in our example, if the umask value is 022 for a user, then we end up with

777 Starting permissions that are assumed for a newly created iNode
-022 Umask value for the user creating the iNode
====
755 Resulting permisions for the iNode

Again, the value of 755 would break back down to the following individual permissions:

For the user: 7 = rwx = Read, Write, Execute
For the group: 5 = r-x = Read, Execute
for everyone: 5 = r-x = Read, Execute

Note:
The default value for us on our linux system at PCC is set to 077 which by default marks files as only usable (read,write,execute) by the owner themselves. You can also use the umask command with the -S option to give symbols to each of the clusters as defined above such as:

```
> umask -S
u=rwx,g=,o=
```

Which illustrates, that only the users have read,write,execute and all members of the group and everyone have no privileges.

==
Question 11
==
Assume that you have two files, f1 and f2, that you want to concatenate and store in a file called f1nf2. Show the cat command needed to perform this task. Now show the cat command needed to put your name, taken from the keyboard, as the last line (concatenate) of f1nf2.

---------------------------------- Answer -----------------------------------
In order to combine or concatenate two files f1 and f2, we can simply specify all the files we want on the command line and and redirect the results to a new file f1nf2. To do this we would enter:

```
> cd
> echo "File1" > f1
> echo "File2" > f2
> cat f1 f2 > f1nf2
> cat f1nf2
File1
File2
```

If we wanted to enhance this operation to include our name taken from the keyboard as the LAST line of this file we would use the command as before, however, we do NOT specify a filename at all wich means to use the keyboard. Instead of using the redirection operator (>), we use the redirection append operator (>>) so that the contents of the file go at the end of the existing file, which in our case contains the contents of f1 and f2:

```
> cat >> f1nf2
John Doe
^C
> cat f1nf2
File1
File2
John Doe
```

Note: We have to enter a Control-C (^C) character to tell the input stream we are done entering values.

lastly, if we wanted to do this all in one command, we can combine both the files and the STDIN all on the same cat command. To specify the STDIN we use the character dash (-) to mean use the STDIN:

```
> cat f1 f2 - > f1nf2
John Doe (joh.doe)
^C
> cat f1nf2
File1
File2
John Doe (john.doe)
```

Intro to Online Learning