Andrew Baca
CS 480
Assignment #4
February 20, 2019

1.1) **chattr**
The chattr command is used to alter attributes of a Linux file system such as adding (+) or removing (-) attributes. There is a set of letters passed as attributes, represented as +-=[aAcCdDeijsStTu], where each individual letter represents various attributes such as things like appending onto a file or backing up a file, for instance, for a file and the +, -, or = sign decides whether to add, remove, or inherit attributes. There are some read only attributes that cannot be modified by chattr.
One option that you may want to use along with the chattr command is the -Verbose option so you can get a little more control of how you want chattr's output. This is also good for looking at what version of chattr you are using.

1.2) **chmod**
The chmod command is used to modify permissions, or change the mode, of files by changing the mode bits. The chmod command's argument's can be passed in various ways along with a file name, such as using octal encoding from 0 – 7, representing binary for the position of the present mode. The other made format with the chmod command is the mnemonic syntax, which used the u-g-a-o specs, representing, user, group, owner, and all, along with other permission chars. These argument mode bits passed along with a file name will change the mode of a certain file or directory by adding or removing permissions to certain files by certain levels of authority.
One option you may want to use along with the chmod command is the -f option, which will suppress most error messages unless they were vital. The reason that this option may be a benefit to use is to key in on vital errors.

1.3) **chown**
The chown command can be used when there needs to be a change made between owner and group of the file. When calling chown, if only an owner is passed as an argument, that user is made the owner of the file, with the group unchanged. If a group is passed as an argument directly after the owner, then chown will change the group as well as the owner. You can also pass a user as the only argument which will change the user to the owner, and the group to the users group. Same case for the group as the only argument with respect to just the group of the file.
One option that you may want to use when calling the chown command is the –from option, which will basically change the owner or group only if its there is a specified match. This is a good form of error or sanity checking

1.4) **getfacl**
The getfacl command can be used to display various attributes of a file ranging from the file name, owners, and permissions to the access control lists of the file. When called, the command will display a nicely aligned output partitioned into 3 sections, with section one displaying filename, owners, users, and groups. The second section displays various ID's and sticky bits. Section three displays user groups and various permissions.
One option that you may want to use along with the getfacl command is the -a option, which will specifically display the access control list of the file. This can be particularly useful when using other commands specifically dealing with other ACL's

1.5) **ln**
The ln command is used to create links in between files. There is four basic forms of the ln command, each creating links between files in different ways. The first form creates a link between TARGET and LINK_NAME, and the second form creates a link between TARGET and the current directory called in. The third and forth forms create link between each TARGET in directory. Both hard links and symbolic links can be crated.
One option that you may want to use when running the ln command is the -t option, which will specify the directory in which to create the links. This is useful in terms of speed so you don't have to cd into a directory prior to ln execution.

1.6) **login.defs**
The login.defs is a text file used to configure the shadow password suite, where each line is a certain configuration parameter with a syntax of a name and a value separated by white space. The configuration parameters can be represented by strings, Booleans represented as yes and no, numbers, and longs. There are several configuration items that are provided within the text file, that configure various aspects of the shadow password suite.
One configuration you may want to in the login.defs file is the CONSOLE which, if defined, root logins will only be allowed on these machines, where as if this is not defined, the root can access any device.

1.7) **lsattr**
The lsattr command can be called to list attributes of a certain file on another extended file system. The lsattr command is associated with the chattr command, and there are various options that can be used with lsattr resulting in various different aspects of the file attributes in different flavors, such as recursively listing attributes, displaying program version, and listing directories.
One option you may want to use with the lsattr command is the -d command, which will list all the directories rather than listing their components.
Another option that may be helpful in terms of readability is the -l option which will print long names opposed to abbreviations.

1.8) **mv**
The mv command is used to move contents of a file into another specified file, basically working as a rename for files, which can also move files from SOURCE to DIRECTORY. If the destination file does not exist, it will be created upon calling mv. There is an array of options to pair with the mv command, resulting in actions such as making backups and ensuring that the given file doesn't get overridden.
If you are dealing with important files while using mv, it would be wise to use the –backup or the -b arguments, which will create a backup of whatever file you are moving.

1.9) **newgrp**
The newgrp command is used to change the current group id while logging in, or logging in to a new group. This command will also try to add the group into the user groupset. If you are not the root user, and you call newgrp, you will be prompted for a password, and will potentially be denied permission if the user is not listed as a group member. You can send an optional -flag to reinitialize the user’s environment. ‘
There are no given options for the newgrp command, however, this command is directly associated with the passwd, shadow, group, and gshadow files.

1.10) **passwd (command)**
The passwd command is used to to change passwords for user accounts. If you are the root user, you can change any users password, however, if you are just a normal user, then you can only change your own password. When this command is called, you must enter the old password correctly in order to successfully change the password. If the user is not permitted to change the password, their request will be denied at this time.
One option you may want to use if you want to use if you want to change your login on the next login is the -e, or –e flag, which will expire an accounts password and force the user to change their password on the next login.

1.11) **passwd (file)**
The passwd file is used to describe the user login accounts for the associated system. You can read permissions allowed for all users here, however, the superuser, or root, is the only user with write access to this file. Each line of the passwd file describes a certain user in seven different fields ranging from name and id's to home directory and shell programs. Root can add new logins in this file. Everything stored in this file is encrypted beyond root.
There are no given options for the passwd file, however, all the field names that can be printed out are name, password, UID, GID, GECOS, directory, and shell.

1.12) **pwconv**
The pwconv command is used to create shadow from passwd, Where entries in the shadow file which don't exist in the main file will be removed. Shadowed entries are updated, and, passwords in the main file are replaced with x. The pwconv command will use values contained in file /etc/login.defs when updating shadow. There are similar commands such as grpunconv command that also update entries from the shadow file based of the main file.
One useful option that can be paired with the pwconv command is the -R, or –root, option which will apply changes and use configuration files from the CHROOT DIR directory.

1.13) **rm**
The rm command is used to remove files from a directory, or remove an entire directory itself. When called, an option may follow, followed by the file or directory that the user would like to remove. There can be more than one file given as arguments. If an interactive mode is is used as an option, and more than 3 arguments are given, then the user will be prompted before the rm operation is completed. One option that can be paired with the rm command is the interactive mode option -I, which will prompt the user if they add more than 3 files before going through with the removal. This can be useful as a sort of sanity check before removing files in bulk.

1.14) **shadow (file)**
The shadowed password file contains the password information for the systems accounts and aging information. Generally, regular users cannot access this file. Each line of the file is an entry containing nine fields such as password and login information like modification dates, expiration dates, reserved fields, and more. The shadowed file is periodically updated with use of the passwd file.
The shadow file does not not include any options, but it does contain various fields within the file related to users, access times and various other times.

1.15) **setfacl**
The setfacl command is used to set file access control lists for files and directories. When called on the command line, a sequence of commands and a sequence of files is listed. The options -m and -x expect an access control list on the command line, while the –set and –set-file options set the access control list of a file or directory. The file owners are granted rights to modify the access control lists.
One option that you may want to pair with the setfacl command is the -k option which will remove default ACL's when called. This can be useful when adding in your custom ACL to reduce overhead.

1.16) **useradd**
The useradd command can be used to create a new user or update default new user information from the command line. When called with the -D option, this command will create a new user account with the additional arguments specified on the command line with additional default values specified by the system. There will also be a default group created for each user that is made.
One option that you would want to use with the useradd command if you have a temporary user that needs to use the system is the -e option, in which you can set a date for this certain user to be disabled.

1.17) **userdel**
The userdel command can be used to delete a user account and their related files from the command line. The named user supplied with the command must exist in the system. If you alter configurations in the login.defs file, you will change the behavior of this command. Some files that the userdel command may alter upon use include passwd, shadow, subgid, and subuid.
One option that you may want to pair with the userdel command is the -r option, which will remove files in the users home directory along with their home directory and the users mail spool. This is useful if you want to erase everything associated with that user.

1.18) **usermod**
The usermod command is used to modify a users account from the command line. Arguments sent along with the command will reflect the changes that are made with the users account. Just as the case with the userdel command, making changes in the login.defs file will alter the use of this command. When using this command, you must be certain that no processes are currently running by the called user. Most Linux engines, if not all of them, will check this case.
One option that you may want to use with the usermod command is the -m option, which will move the contents of one users home directory into a new home directory. This can be useful if you want another user to inherit files from a user.

1.19) **vipw**
The vipw command can be used to to edit the password, group, shadow password or shadow group file. If the -s flag is used with the vipw command, then the shadow versions of the files will be updated. The program will make appropriate locks to deal with file corruption. For an editor, the command will first try to access $VISUAL, then it will try to access $editor, and will then go to a default editor if both those fail.
One option you may want to use with the vipw command is the -p command if you are specifically trying to edit the passwd database.

2)
The purpose of shadow password files is to store users encrypted passwords so that whenever there is a security threat to the system, these encrypted passwords aren't available to them. The shadow password

file is only readable by the superuser. The shadow password file also includes account information, spanning nine fields separated by colons containing login names, dates of password changes, between password changes, and various other bits of information pertaining to users passwords, one line per user. The shadow password files are usually default on all systems. Both the passwd file and the shadow password file must be maintained and this cone be done manually, or with the help of various tool such as useradd.

3) Skipped on behalf of Instructor

4) A user's default group is determined based on the group ID, made in /etc/group associated with the user and can be accessed to change when vipw is called. Generally, when a new user is created, a default group of users with the group id of 100 will be assigned to that new user, unless specified at creation. In order to set their default ID, you can go into the /etc/group file, and add the username after the group information. From here, you can call vipw, and alter the group ID section to the default group ID that matches the group folder.
A good way to check that the group ID is the one you want it to be is to call yast, and go to user configuration, and see if the group ID is the wanted one there.
Files In tar.bz2:

- group_copy
- passwd_copy

5) In order to create a new user without using any high level program such as yast, useradd, groupadd, we go to the passwd file, and add the user john into it, and this can be done by calling vipw to access the passwd file. We call passwd for user for john (passwd jclass) in order to create the password for john, which we set to **john12345**. Next, we copied the /etc/skel directory into the /home/jclass directory using the cp command with the -Rv flag, copying the skeleton files into /home/jclass to get the pertinent files. Finally I called the chown command with the -R option it use recursion and change ownership of all files from root root to jclass users. I also called chmod -R u=rwx jclass to change the permissions of jclass as well.
To test if the user jclass was implemented correctly, I logged out of root, and logged into jclass. From here, I did various terminal commands to list and access files to make sure that everything was copied over properly and the ownership were changed.
Files in tar.bz2:

- passwd_copy
- jclassOutputPerms.txt

6) In order to make it so that /hw4playground/tmp to where every user can write into this directory, but not delete files from it, we called chmod +t tmp within the directory to set the sticky bit on this file.
In order to change the permissions so that only barbara and root can create files and only barbara and ntadmin can list and access files, I used the command setfacl -m u:barbara:rwx barbara, which added the access control list for barbara to have access to read, write, and execute programs within /hw4playground/barbara.
For ntadmin to have permissions to list and execute programs, I called the setfacl command again, in this syntax: setfacl -m g:ntadmin:rx barbara, which gave group ntadmin permission to read and execute files, however, they cannot write into the files.
To test barbara's access to the /hw4playground/barbara file, I logged out of root, logged into barbara, and did various file operations to make sure all the functionality granted works as promised.
Files in tar.bz2:

- prob6OutputPerms.txt

7) I first made a Perl script that would call pwd and store it as a mount point, making the assumption that the file system was mounted in whatever directory that this script was ran from. Next, I called ls -li to find the which will print an extended ls -ls of file information, ans I specifically extracted the inode numbers into array @lsInput. I piped the ls -li command with awk '{print}' to just extract the inode numbers out of the listing of file information. Next, I repeatedly called the find command within a while loop using the mount point, and using the -xdev option to not cross over into any other file-systems, followed by the -inum flag and the inum to give the find command an inode to a hard link. This script will just look for hard links within a present working directory making the assumption that the present working directory is the mount point at all times.
This code resulted in printing out the hard link matches for inodes within this mounted file system in a directory.
This method is discussed on page 136 of the Linux admin book, covering how to find hard links with inodes and mount points.
Code can be looked at in findLinks.pl in the tar file. nf.link, test, test2, and test2link are all files and their hard links that I had created to test this program within this directory.
Files in tar.bz2:

- findLinks.pl
- nf.link
- test2
- test2link
- test
- problem7out.txt

8) I started this problem by first calling the mkdir command with an argument cs480home attached to it in the root directory, creating the directory /cs480home in root.
In order to change the default directory that new users will be created in when the useradd command is called, I went to the /etc/default/useradd file and changed the HOME variable to equal /cs480home, meaning that whenever there is not a proper option supplied with a given home directory, /cs480home will be the default directory when useradd is called.
In order to test the validity of the default home directory when useradd is called, I created a user called hw4user with the useradd command, then called passwd and assigned **hw4321** to be hw4user's password. I logged onto hw4user and called pwd to verify that the user was in the wanted home directory.
Files in tar.bz2:

- useradd_copy
- passwd_copy

Feedback:
Time Breakdown:

1)	3 hours
2)	30 mins
4)	1 hour
5)	1.5 hour
6)	1 hour
7)	2 hours
8)	1 hour
cleanup/doc)	2 hours

TOTAL: 12 hours

Overall I thought that this assignment was fun to do. It is cool to see the low level ways to create users and give them permission to certain resources in the system. Being able to change small details of passwd files through login.defs was one particular bit I found interesting. One question that I had some confusion with was the scripting question, particularly pertaining to the mount point and the file system to traverse to find the hard links.