Andrew Baca
CS 480
Assignment #2

1.1) **bzip2**
The bzip2 command compresses files using both Huffman Encoding as well as the Burrows Wheeler text sorting algorithm. The bzip2 command is followed by a list of filenames and corresponding flags, and when called, will replace the file with a compressed version titled “original_name.bz2", where the copies share information such as permissions and modification dates so all the properties can be properly restored upon decompression. The bzip2 command will not overwrite the existing files unless you set the -f flag. You can use the bunzip2 command or the bzip2 -d command to decompress given files. Whenever the command is used, compression is always performed, even if the original file is smaller than the compressed file which is usually the case when the original file is less than100 bytes in size.
-v –verbose is an option that will show compression ratios between the original files and the compressed files. This command would be particularly useful for somebody who compresses files a lot who wants to see if there is a more efficient way to manage space, or whether you would be saving space or not by compressing the files.

1.2) **find**
The find command is used to search the file tree structure for a given target, whether it be a certain directory, a certain filename, or certain permissions granted to certain files or directories. The find command starts the directory tree traversal at a specified starting point, and if none are specified, then ‘.’ is assumed, meaning the whole tree. The -H, -L, and -P options deal with the symbolic links, pertaining to whether symbolic links should be followed or not. If you were to use the -D debugoptions, this will print diagnostic information as the directory tree is being traversed.
One option that you may want to use if you were investigating a crime scene, and a Linux laptop was a primary piece of evidence is the -atime n command, so you can see exactly which files have been accessed. If you thing a file has been tampered with during the crime, you would want to use the -ctime command, which will give the investigator which files have been changed under a specified amount of time.

1.3) **grep**
the grep command can be used to find patterns in a given file, or working directory, and prints the matching lines. The patterns being searched for can be represented using regular expressions, which is an expression that can represent a pattern in a string. Grep understands three different syntax’s of regular expressions and they are basic, extended, and perl. Some common options within the grep command include -c, which prints out the number of lines that match a given pattern, as well as -l which will list the file names that contain the matching pattern, and -n , which will display the matching lines and their respected line numbers.
One option that you may want to use -H if you are only interested in what files contain the pattern you are searching for while using grep command. You may want to delete all the files that have a certain persons name in a given directory, so you would use the grep command with the -H option to find all the files to delete.

1.4) **kill**
Kill is the command used to terminate a process, and when it is called, the kill command sends a specified signal to process(es). The default signal is called TERM, and it is used to terminate a process if no signal is specified. When kill is called, the processes specified can be specified by both names

and process ID's. The kill command has three potential return codes, and they are 0, representing a success, 1, representing a failure, and 64, representing a partial success. Some common options within the kill command include -s, which is a signal to send given as a name or number, and -l, which will print a list of signal names.
One command you may want to use if you are just looking for the Process ID's of the processes being killed is the -p –pid command, which will just print the PID's and not send any signals.

1.5) **lpq**
The lpq command shows the current print queue status on a specified printer. Printing jobs will be queued on a default destination if there is no printer specified. If you are to use the +interval option with the lpq command, you will continuously get feedback pertinent to the jobs in the queue until its is empty. Some common options pertinent to lpq are -E, which forces encryption when the server is being connected, -P, which specifies an alternate printer, -U, which specifies an alternate username, and -h to specify an alternate server.
A lab administrator may want to use the -a option to see the jobs on all printers. He or she would use this command as sort of a diagnostics check.

1.6) **lprm**
lprm is a command used to cancel cancel print jobs that are in the queue and waiting to be printed. If there are no arguments that are given with lprm, then the default would be to cancel the current job on the default destination. You can specify more than one job to be canceled with this command. Some common options to go along with the lprm command include -E, which forces encryption, -P, which specifies printer destination, and -U, which specifies an alternate username.
System administrators may want to use the -P option to specify a printer destination if there is multiple printers in the facility that they are administrating.

1.7) **ls + Permissions**
the ls + permissions command List information about the files, which are sorted alphabetically, within a given directory. The ls + permissions command has an extensive list of options, which will output various bits of information attached to each file within the directory, such as the name of the author, size of the file, list of the directories themselves, modification time, and format.
Somebody may want to use the –author option to find out the creator of each file within a given directory. A system admin may want to use the -d option with the ls command to see a full list of the directories within.

1.8) **man**
the command man refers to an interface within the terminal to the on-line reference manuals. Each page argument that is given to the man command is the name of some program, utility, or, function within your system. When an argument is paired with the man command, a neatly formatted document with descriptions of certain functions and their attributes are printed for the user to scroll. Man pages are broken into sections containing different functionalities of the given system, and all the information on how to use that tool.
A new computer science major may want to use man pwd to learn basic maneuverability of their terminal and directories, as well as man cd, so they can traverse through the directory tree.

1.9) **scp**
The command scp can be used to securely copy files between a host and a network. Scp uses ssh(1) for data transfer and other aspects of security, and when called, scp will ask for passwords for authentication. File names possibly contain user and host specification to indicate where the file is to

be copied to. Some of the common options within the scp command include -3, which copies between 2 remote host connected through local host, -4, which forcesIPv4, -6, which forces IPv6, and -C, which enables compression.
If you are concerned about a certain aspects of the SSH encryption technique to use with the scp command, you may want to use the -c cipher command, as well as look into the -F ssh config command.

1.10) **script**
the script command makes a typescript of everything that is displayed on ones terminal. It is particularly useful whenever a copy of the work done in the terminal is needed. If there is a filename given as an argument, then script saves the terminal dialogue into this file. The default file that the script command will write the display into will be titled typescript if no argument is given. Script is terminated by the exit command. Some options that are included within the display command a –append, which will append to a file, or typescript, -c –command, which will run the command rather than the shell, -e – return, which will return the exit code of the child process, and -f –flush, which will flush the typescript after each use.
One option that may be useful to somebody running the typescript command often might be flush. This would help the user because they wnt have to append through older terminal data every view.

1.11) **ssh**
The ssh command is used to access a program used for logging onto a remote machine and for execute commands on that machine, using methods of secure communications between two insecure host. When executing ssh, the user connects and logs onto a host name, and must prove their identity using various methods including public key authentication. Ssh may get information from a configuration file on the user and system level. There are various methods of authentication that are arranged in in an order of preference.
A person may want to use -X forwarding to bypass certain file permissions when using the ssh command to link to another system, although this can lead to security vulnerabilities such as keystroke monitoring. Alternatively, -x Disables the X11 forwarding.

1.12) **su**
The command su allows the user to execute commands through a different user and a group ID. The default argument for the su command is root, where you will be asked for a password for authentication. With the su command, you can basically switch users if you have the valid login credentials.
One option you may want to use with the su command would be -login if you are trying to prevent mixing of environments and want a login environment similar to a real login.

1.13) **sudo**
the command sudo allows you to execute a command as another user, possibly a user of higher authority. Once executed, the user has to go through authentication, and then is allowed to execute command under that user. The authentication of sudo requires that the users enter their passwords within a certain time span, or it will exit if the time limit is reached. As a security precaution, administrators may log ones attempts to use the sudo command.
One option that you may want to use while executing sudo is the -E which means the user requested permission to keep the same environment variables, and it is up to the security policy to allow this or not.

1.14) **sudoers**

the sudoers command is a security policy that determines the privileges of sudo. Some common policy in this plugin states that users must use a password to authenticate themselves when the sudo command is called. If a user tries to run sudo command who is not on a list of acceptable users, mail must be sent to the administrator regarding the command. There are certain mail bypass flags that can make the proper authority unaware about the sudo execution from a user who is not approved. All attempts to run sudo are logged no matter what.
Sudoers has an extensive set of Boolean options. One might consider running sudoers with the authenticate option if the user would like to identify themselves via password before running commands.

1.15) **tar**
The command tar is an archiving tool used to store multiple files into a single file. Options to the tar command can lead to three different styles, and they are traditional style, unix or short option style, and gnu or long option style. These different styles are represented as different commands on the terminal. Ideally the tar command always has one option or more linked to it.
One may want to use the -A option to concatenate the archived file to the end of another. This would be good for an ongoing project or an ongoing data set.

1.16) **yast**
YaST is a command used various aspects of a system such as hardware, network connections, network clients and services, and other system options. There are three potential front ends that will be selected upon when YaST is called, and the front end is automatically selected from either GTK, QT, or ncursed based on the available components in the environment you are working in. If you call YaST by itself, the YaST2 control center will appear.
One option that you may want to use when using the yast command is the –qt or the –gtk command, depending on an interface preference that you would like to use, the QT interface, ot the GTK interface, QT being more graphical with icons.

1.17) **zypper**
The zypper command will lead to the zypp system management library when called, which is used to install, update, and remove software, as well as manage repositories and perform various queries. System packages are the set of installed packages on a system, which can only be deleted, and not added. The Zypp library has the ability to add, delete, and manipulate repositories using the commands found in the repository management section of the zypper command.
You may want to use the info option if you to read about information regarding specific packages, and see the bast possible version available.
You may want to use the rr command to remove any repositories that are not needed within the Linux system.

2)
In order to view the repositories listed with the URI included, you would want to use the zypper lr -d command. This command will print information about the repositories such as alias, name, whether it is enabled, GPG check, refresh, priority, the type of service, and the URI.
From here, I checked which URI contained https, and removed those repositories using the zypper rr # command, with the # specifying the number of repository to delete.
Once the unwanted repositories were deleted, we use an scp (secure copy) command in order to copy the missing man pages from the host computer to our virtual machines. The command used to do so was scp -r abaca@lbc:/tmp/CS480/ /usr/share/man.

Lastly, I finally installed the missing man pages onto my VM using the zypper install usr/share/man/*.rpm, and continued to test to see if they were installed using the man intro and man 3p write command and options.

3.) In order to successfully create the users for the VM, I used the yast command to pull up the YaST interface. From here, I went to the securities and users tab, and added the users Jim, Joe, Bob, Bill, and Barbara. The username and passwords are as follows:

jim	jim12345
joe	joe54321
bob	bob12345
bill	bill54321
barbara	barbara12345

Once I created the users, I went on to the sudoers file in order to bypass root password authentication for servicing the printer for jim and joe, as well as kill and reboot for bob, bill, and barbara. This was done by first adding two sets of User_Alias specifications for jim and joe to service the printers, and bill, bob, and barbara to kill and reboot processes. Code segments:

```
User_Alias      PRINTERS = jim, joe
User_Alias      KILLERS = bob, bill, barbara
```

Next, I added the Cmnd Alias Specifications with the commands to service, kill, and reboot the printer. This was done with the code:

```
Cmnd_Alias    Service = /usr/bin/lprm, /usr/sbin/lpc
Cmnt_Alias    KILL = /sbin/reboot, /bin/kill
```

Then I deleted the two lines that the comments within sudoers said to delete, and I added the User privilege specification for the printers and killers:

```
PRINTERS ALL = SERVICE
KILLERS ALL = KILL
```

Next, I exited sudoers, logged out of root, and went on to log into jim, joe, bob, bill, and barbara to see if I bypassed the root password for the given tasks.

4.) The code can be found in the bz2 compressed file in the root */home/CS480/abaca*
For this problem, I made a perl code that reads through the /etc/passwd and /etc/group and finds UID's and GID's that match between the files, and for each user, I printed the id match as well as the group the user is in.
This was done using two while loops, with the outer loop pulling out the UID, and comparing it to all the group ID's in the inner while loop. The first file is opened outside both loops and closes outside both loops, while the second file is opened and closed before and after each outer loop execution. This was done so that one user ID can be compared to all group ids to find a match. Once the match was found, the results were displayed.
5) In order to locate suid files on my system, I had to use the find command with the permission options set with the argument Making sure the permission bit was set for both the users and the groups. This can be achieved using the commands:

```
find / -perm /u=s
find / -perm /g=s
```

Some of the files that were displayed after using these commands included:

- /usr/bin/sudo is a file where the setuid functionality needs to be correct because this is the structure of the whose permissions set of the system. If this is not functioning properly, the whole permission structure of the system, and the system in general will be at risk of unwanted attacks, privacy and other problems

- /usr/bin/expiry is a file that deals with password expiry information, so it wouldn't be good to alter or have the setuid functioning improperly for this file.
- /usr/bin/passwd file can be used for the users to alter their passwords, and it is vital for the setuid to not be altered and functioning properly on this, posing major security risk if not.
- /sbin/unix2_chkpwd is used in application for checking passwords, so the setuid on this would cause problems if altered, missing.
- /sbin/unix_chkpwd is also used for password checking, so you wouldnt want to mess with the setuid .

6) A files real UID is an id number inherited from the user who created the file as a copy. The EUID, also known as the effective UID is an extra UID that determines what resources this certain file can use and what permissions that this file has. For most processes, UID and EUID are basically used the same, except for programs that involve setuid. The main reason why both of these forms of id exist is because their needs to be somewhat of a distinction between identities and permissions. They are basically the same, with EUID being an extension of UID.

Assignment Feedback:

Overall this assignment was very time consuming, and there was some confusing points in the directions such as the execution of number 5, however, toward completion I really felt like I learned a great deal. I am far more comfortable with man pages that I was prior to the assignment, and I feel like I have a whole new arsenal of Linux commands that will be helpful to be in the future. I learned a bit of Perl, which I feel may be the friendliest language to use when it comes to reading from files and doing basic parses. Number 1 was an extremely time consuming problem, however, I feel my understanding of essential Linux commands, terminal navigation, and man page understanding in terms of vocabulary and structure has greatly improved.

Overall Time/Breakdown

Report, organization, preparing	1 hour
#1	4 hours
#2	1 hour
#3	3 hours
#4	2 hours
#5	1.5 hour
#6	.5 hours
Total:	13 hours