Lab No. 03

Controlling Access to Files, and Managing Packages using Commands

Objective:

This lab will introduce the basic concept of file access permissions and package management in Linux to you.

Activity Outcomes:

On completion of this lab students will be able to:

- Reading and setting file permissions.
- Setting the default file permissions.
- Performing package management tasks.

Instructor Notes

As pre-lab activity, read Chapter 09 and 14 from the book "The Linux Command Line", William E. Shotts, Jr.

1) Useful Concepts

File Permissions

Linux is a multi-user system. It means that more than one person can be using the computer at the same time. While a typical computer will likely have only one keyboard and monitor, it can still be used by more than one user. For example, if a computer is attached to a network or the Internet, remote users can log in via ssh (secure shell) and operate the computer. In fact, remote users can execute graphical applications and have the graphical output appear on a remote display. In a multi-user environment, to ensure the operational accuracy, it is required to protect the users from each other. After all, the actions of one user could not be allowed to crash the computer, nor could one user interfere with the files belonging to another user.

id command

In the Linux security model, a user may own files and directories. When a user owns a file or directory, the user has control over its access. Users can, in turn, belong to a group consisting of one or more users who are given access to files and directories by their owners. In addition to granting access to a group, an owner may also grant some set of access rights to everybody, which in Linux terms is referred to as the world.

User accounts are defined in the /etc/passwd file and groups are defined in the /etc/group file. When

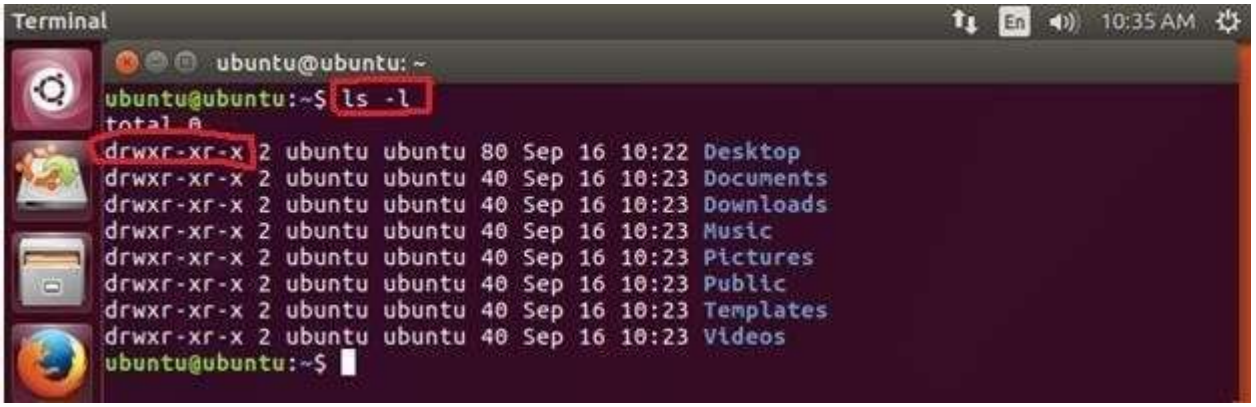


user accounts and groups are created, these files are modified along with /etc/shadow which holds information about the user's password.

Option	Explanation
-g	Print only the effective group id
-G	Print all Group ID's
-n	Prints name instead of number.
-r	Prints real ID instead of numbers.
-u	Prints only the effective user ID.

Reading, Writing, and Executing

Access rights to files and directories are defined in terms of read access, write access, and execution access. If we look at the output of the ls command, we can get some clue as to how this isimplemented:



The first ten characters of the listing are the file attributes. The first of these characters is the file type. Here are the file types you are most likely to see:

Attribute	File Type
-	A regular file.
d	A directory.
l	A symbolic link.
c	A character special file. This file type refers to a device that handles data as a stream of bytes, such as a terminal or modem.
b	A block special file. This file type refers to a device that handles data in blocks, such as a hard drive or CD-ROM drive.

The remaining nine characters of the file attributes, called the file mode, represent the read, write,and execute permissions for the file's owner, the file's group owner, and everybody else.

User	Group	World
rwx	rwx	rwx

Attribute	Files	Directories
r	Allows a file to be opened and read.	Allows a directory's contents to be listed if theexecute attribute is also set.

w	Allows a file to be written	Allows files within a directory to be created, deleted, and renamed if the execute attribute is also set.
x	Allows a file to be treated as a program and executed.	Allows a directory to be entered, e.g., cddirectory.

For example: **-rw-r--r-** A regular file that is readable and writable by the file's owner. Members ofthe file's owner group may read the file. The file is world-readable.

Reading File Permissions

The ls command is used to read the permission of a file. In the following example, we have used ls command with -l option to see the information about /etc/passwd file. Similarly, we can read the current permissions of any file.

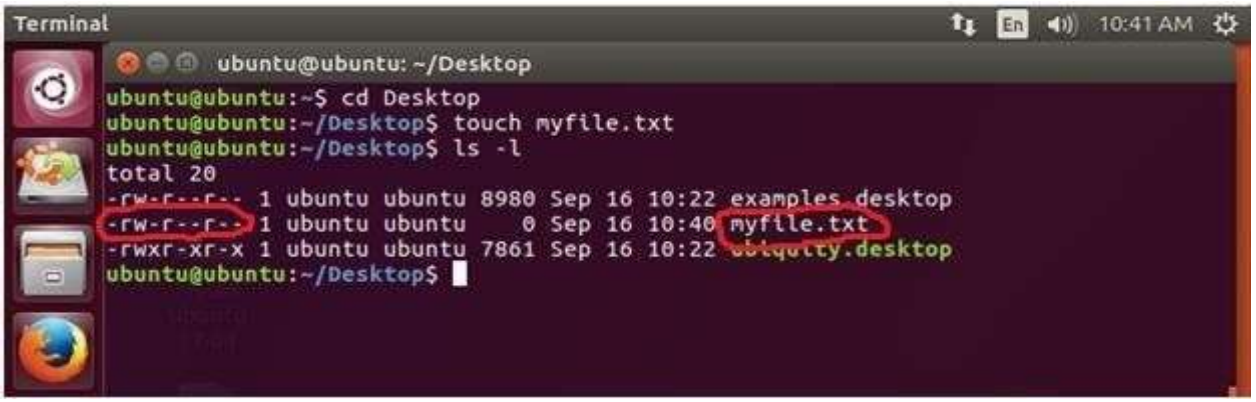


Change File Mode (Permissions)

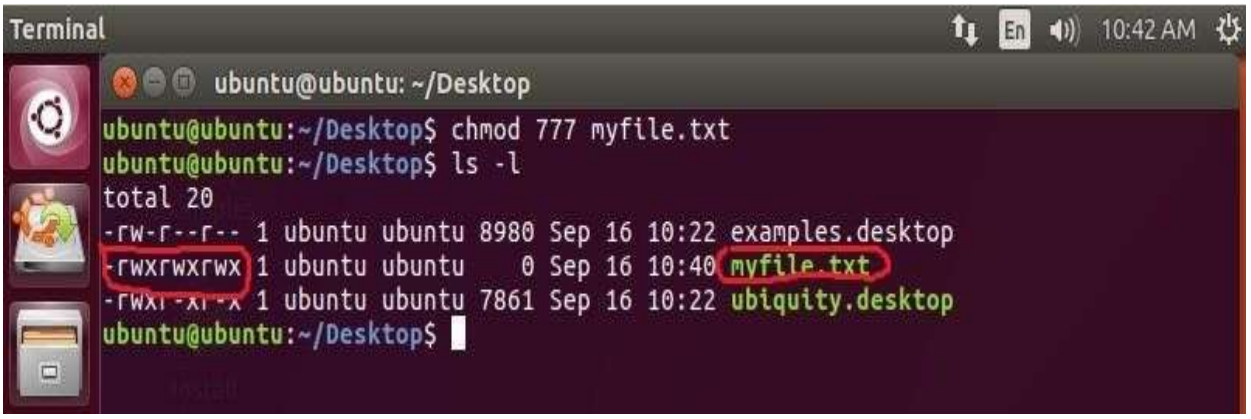
To change the mode (permissions) of a file or directory, the **chmod** command is used. Beware that only the file's owner or the super-user can change the mode of a file or directory. **chmod** supports two distinct ways of specifying mode changes: octal number representation, or symbolic representation. With octal notation we use octal numbers to set the pattern of desired permissions. Since each digit in an octal number represents three binary digits, these maps nicely to the scheme used to store the file mode.

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

In the following example, we first go to the Desktop directory using cd command. In Desktopdirectory, we create a text file **"myfile.txt"** using touch command and read its current permissions using ls command.



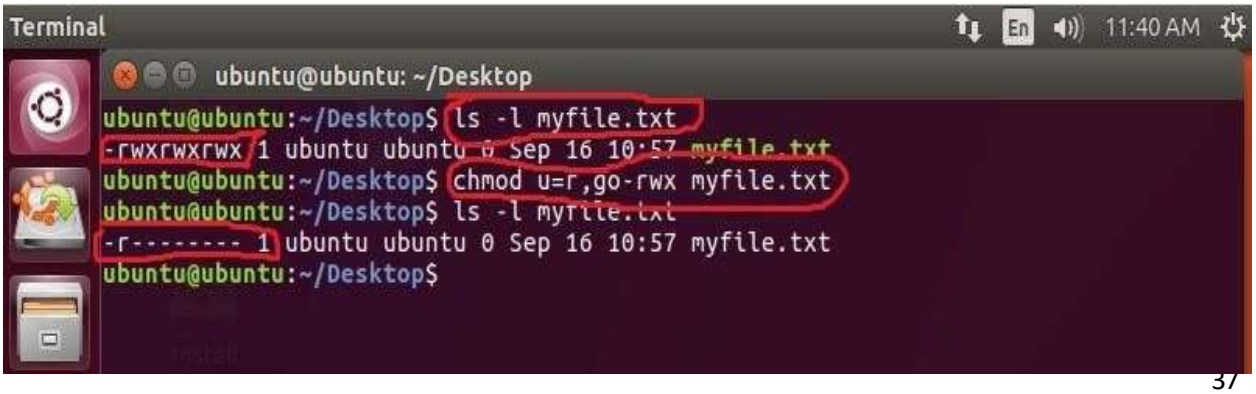
Now, we change the permission of myfile.txt and set it to 777 that is everyone can read, write and execute the file.



chmod also supports a symbolic notation for specifying file modes. Symbolic notation is divided into three parts: who the change will affect, which operation will be performed, and what permission willbe set. To specify who is affected, a combination of the characters “u”, “g”, “o”, and “a” is used as follows:

Character	Meaning
u	Owner
g	Group
o	Others
a	all

If no character is specified, “all” will be assumed. The operation may be a “+” indicating that a permission is to be added, a “-” indicating that a permission is to be taken away, or a “=” indicating that only the specified permissions are to be applied and that all others are to be removed. For example: **u+x,go=rx** Add execute permission for the owner and set the permissions for the group and others to read and execute. Multiple specifications may be separated by commas. In the following example, we change the permissions of myfile.txt using symbolic codes. As all of the permissions of myfile.txt were



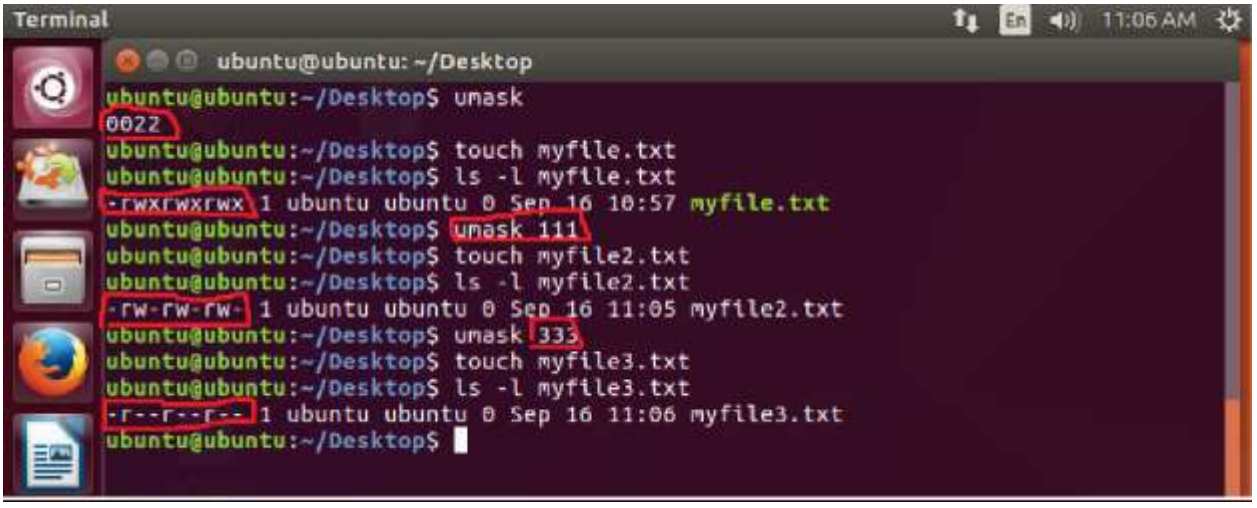
set previously, now we make it readable only to the user while the rest cannot read, write or execute the file.

Controlling the Default Permissions

On Unix-like operating systems, the umask command returns, or sets, the value of the system's file mode creation mask. When user create a file or directory under Linux or UNIX, he/she creates it with a default set of permissions. In most case the system defaults may be open or relaxed for file sharing purpose. umask command with no arguments can be used to return the current mask value. Similarly, If the umask command is invoked with an octal argument, it will directly set the bits of the mask to that argument.The three rightmost octal digits address the "owner", "group" and "other" user classes respectively. If fewer than 4 digits are entered, leading zeros are assumed. An error will result if the argument is not a valid octal number or if it has more than 4 digits. If a fourth digit is present, the leftmost (high- order) digit addresses three additional attributes, the setuid bit, the setgid bit and the sticky bit.

Octal Value	Permissions
0	read, write, execute
1	read and write
2	read and execute
3	read only
4	write and execute
5	write only
6	execute only
7	no permissions

In the following example, we first read the current mask that is 0022 and then we created a file myfile.txt using this mask and display its permission. Then we reset the mask with 111 and 333 octal values and create new files. It can be seen clearly that new files are created with different default



permissions.

Changing User Identity

At various times, we may find it necessary to take on the identity of another user. Often, we want to

gain superuser privileges to carry out some administrative task, but it is also possible to “become” another regular user for such things as testing an account.

Run A Shell with Substitute User and Group IDs

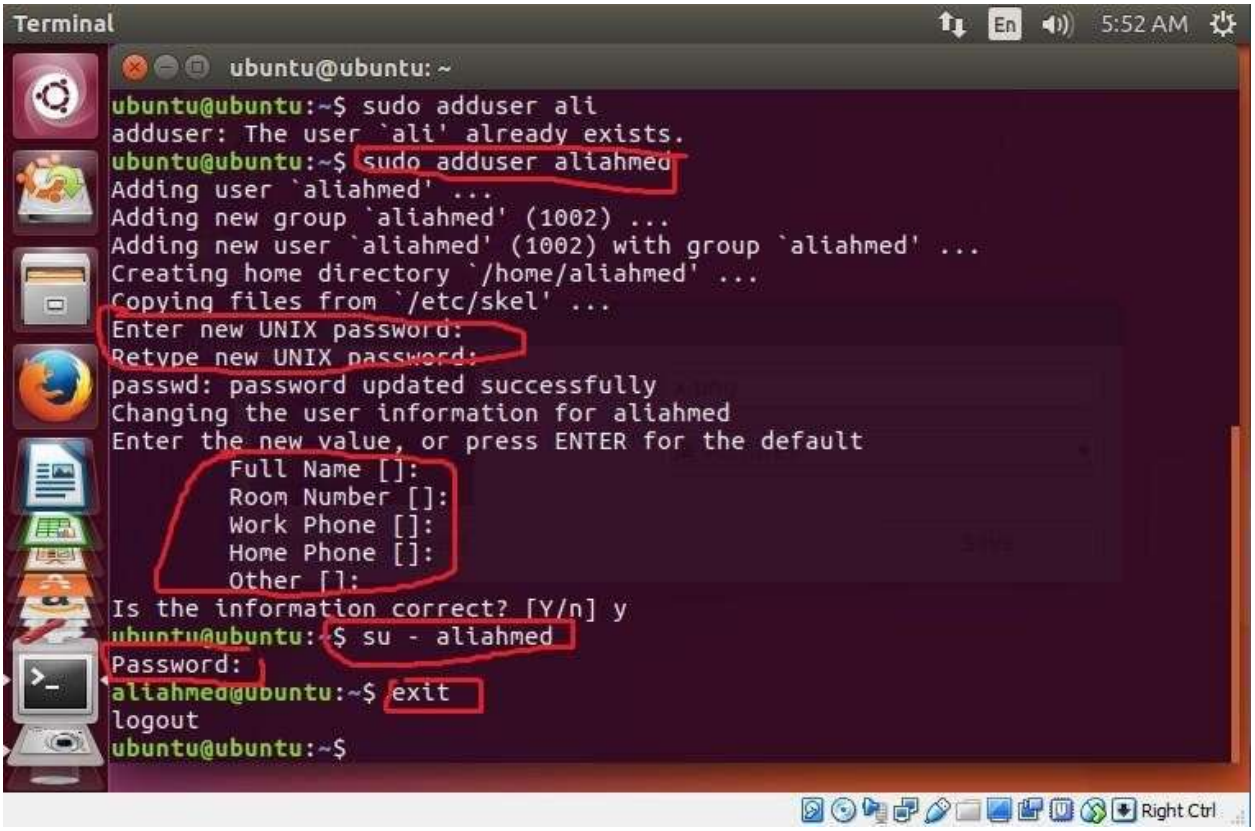
The su command is used to start a shell as another user. The command syntax looks like this:

```
su -l username
```

Execute A Command as Another User

On Unix-like operating systems, the sudo command ("superuser do", or "switch user, do") allows a user with proper permissions to execute a command as another user, such as the superuser.

Example: In the following example first, we created a new user “aliahmed” using adduser command. Then we run the shell as aliahmed. In the end we logout using exit command.

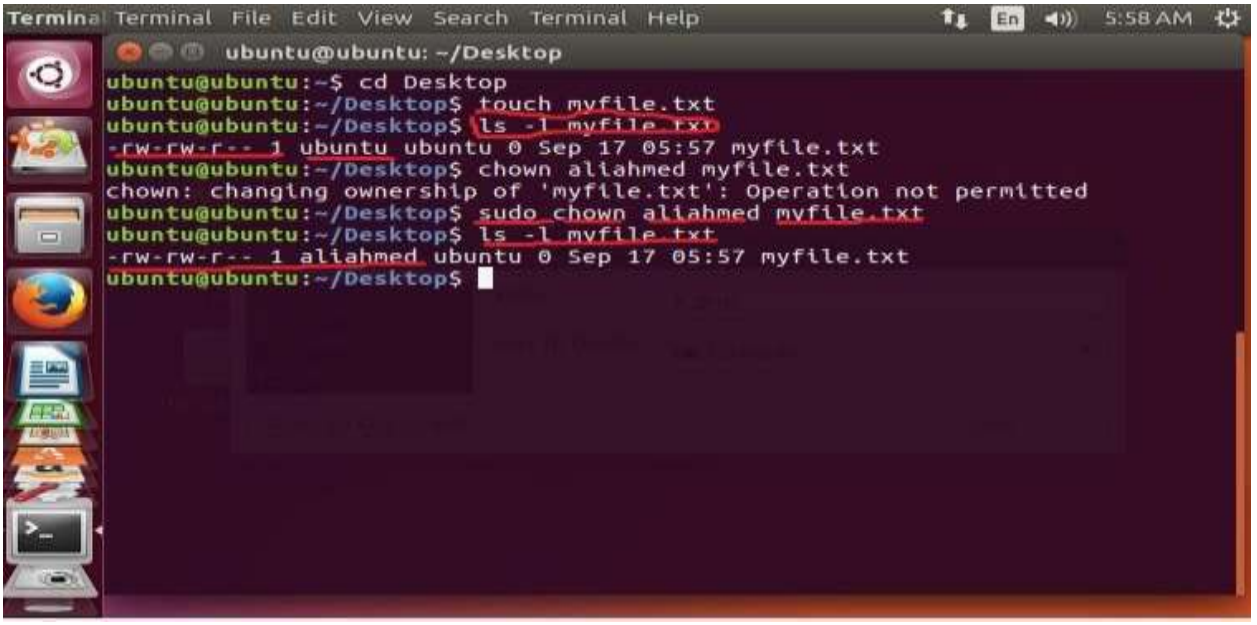


Change File Owner and Group

The chown command is used to change the owner and group owner of a file or directory. Superuser privileges are required to use this command. The syntax of chown looks like this:

```
chown owner:group file/files
```

Example: In the following example first, we created a file named myfile.txt as user ubuntu. Then we changed the ownership of myfile.txt from ubuntu to aliahmed.



Package Management

Package management is a method of installing and maintaining software on the system. Linux doesn't work that way. Virtually all software for a Linux system will be found on the Internet. Most of it will be provided by the distribution vendor in the form of package files and the rest will be available in source code form that can be installed manually.
Different distributions use different packaging systems and as a general rule, a package intended for one distribution is not compatible with another distribution. Most distributions fall into one of two camps of packaging technologies: the Debian ".deb" camp and the Red Hat ".rpm" camp. There are some important exceptions such as Gentoo, Slackware, and Foresight, but most others use one of these two basic systems.

Package Files

The basic unit of software in a packaging system is the package file. A package file is a compressed collection of files that comprise the software package. A package may consist of numerous programs and data files that support the programs. In addition to the files to be installed, the package file also includes metadata about the package, such as a text description of the package and its contents. Additionally, many packages contain pre- and post-installation scripts that perform configuration tasks before and after the package installation.

Repositories

While some software projects choose to perform their own packaging and distribution, most packages today are created by the distribution vendors and interested third parties. Packages are made available to the users of a distribution in central repositories that may contain many thousands of packages, each specially built and maintained for the distribution.

Dependencies

Programs seldom "standalone"; rather they rely on the presence of other software components to get their work done. Common activities, such as input/output for example, are handled by routines shared

by many programs. These routines are stored in what are called shared libraries, which provide essential services to more than one program. If a package requires a shared resource such as a shared library, it is said to have a dependency. Modern package management systems all provide some method of dependency resolution to ensure that when a package is installed, all of its dependencies are installed too.

High and Low-level Package Tools

Package management systems usually consist of two types of tools: low-level tools which handle tasks such as installing and removing package files, and high-level tools that perform metadata searching and dependency resolution. For Debian based systems low-level tools are defined in dpkg while high-level tools are defined in apt-get, aptitude.

Common Package Management Tasks

Finding a Package in a Repository: Using the high-level tools to search repository metadata, a package can be located based on its name or description. In Debian based systems it can be done as given below:

```
apt-get update
apt-cache search search string
```

Example: To search apt repository for the emacs text editor, this command could be used:

```
apt -get update
apt-get search emacs
```

Installing a Package from a Repository: High-level tools permit a package to be downloaded from a repository and installed with full dependency resolution.
Example: To install the emacs text editor from an apt repository:

```
apt-get update; apt-get install emacs
```

Installing a Package from a Package File: If a package file has been downloaded from a source other than a repository, it can be installed directly (though without dependency resolution) using a low-level tool.

```
dpkg-install package_file
```

Example: If the emacs-22.1-7.fc7-i386.deb package file had been downloaded from a non- repository site, it would be installed this way:

```
dpkg –install emacs-22.1-7.fc7-i386.deb
```

Removing A Package: Packages can be uninstalled using either the high-level or low-level tools. The high-level tools are shown below.

```
apt -get remove package_name
```

Example: To uninstall the emacs package from a Debian-style system

```
apt -get remove emacs
```

Updating Packages from a Repository: The most common package management task is keeping the system up-to-date with the latest packages. The high-level tools can perform this vital task in one single step.
Example: To apply any available updates to the installed packages on a Debian-style system:

```
apt -get update; apt-get upgrade
```

Upgrading a Package from a Package File: If an updated version of a package has been downloaded from a non-repository source, it can be installed, replacing the previous version:

```
dpkg --install package_file
```

Listing Installed Packages: These commands can be used to display a list of all the packages installed on the system:
`dpkg --list`

Determining If A Package Is Installed: These low-level tools can be used to display whether a specified package is installed
`dpkg --status package_name`
Example:
`dpkg --status emacs`

Displaying Information About an Installed Package: If the name of an installed package is known, the following commands can be used to display a description of the package:

```
apt -cache show package_name
```
Example:
```
apt -cache show emacs
```

2) Solved Lab Activities

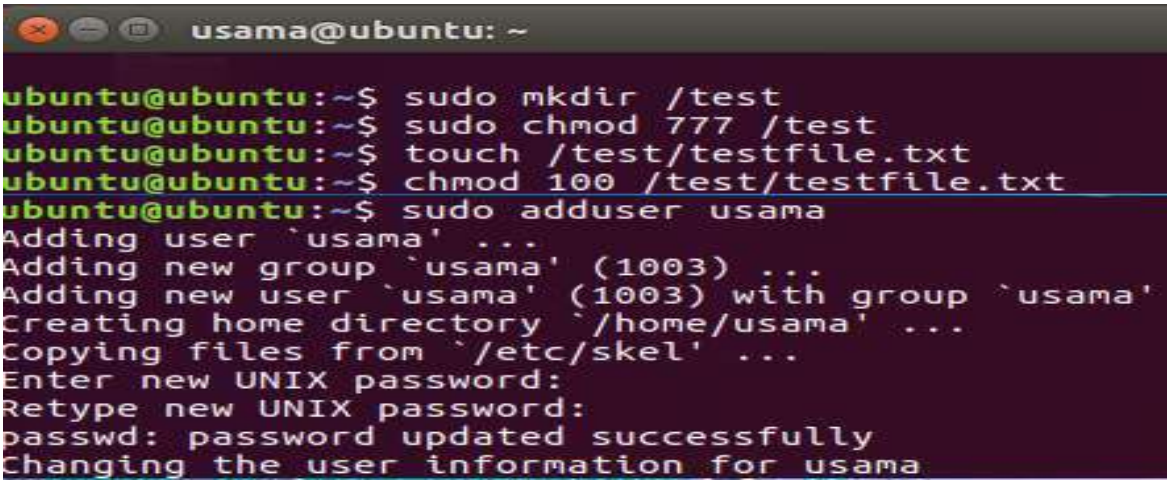
Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
1	*15*	*Medium*	*CLO-5*
2	*15*	*Medium*	*CLO-5*
3	*15*	*Medium*	*CLO-5*

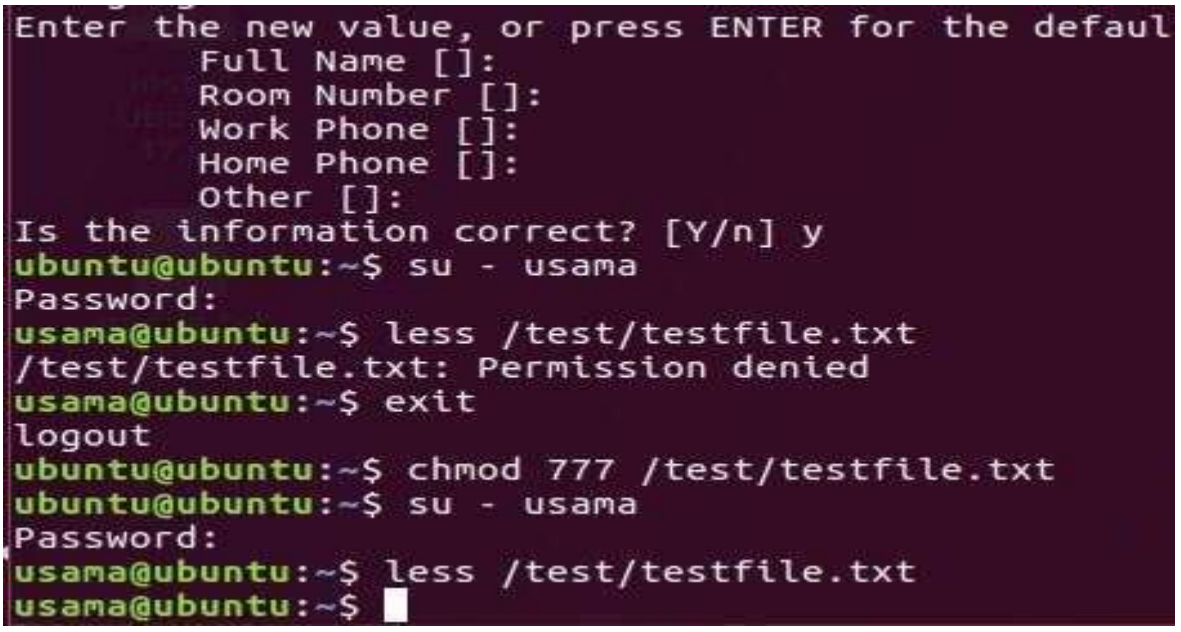
Activity 1:

This activity is related to file permission. Perform the following tasks

- ***Create a new directory named test in root directory as superuser***
- ***Make this directory public for all***
- ***Create a file "testfile.txt" in /test directory***
- ***Change its permissions that no boy can write the file, but the owner can read it.***
- ***Create another user "Usama"***
- ***Run the shell with user Usama***
- ***Try to read the "testfile.txt"***
- ***Logout as Usama***
- ***Change the permission of testfile.txt so that everyone can read, write and execute it***
- ***Run shell as Usama again***
- ***Now, read the file***

Solution:



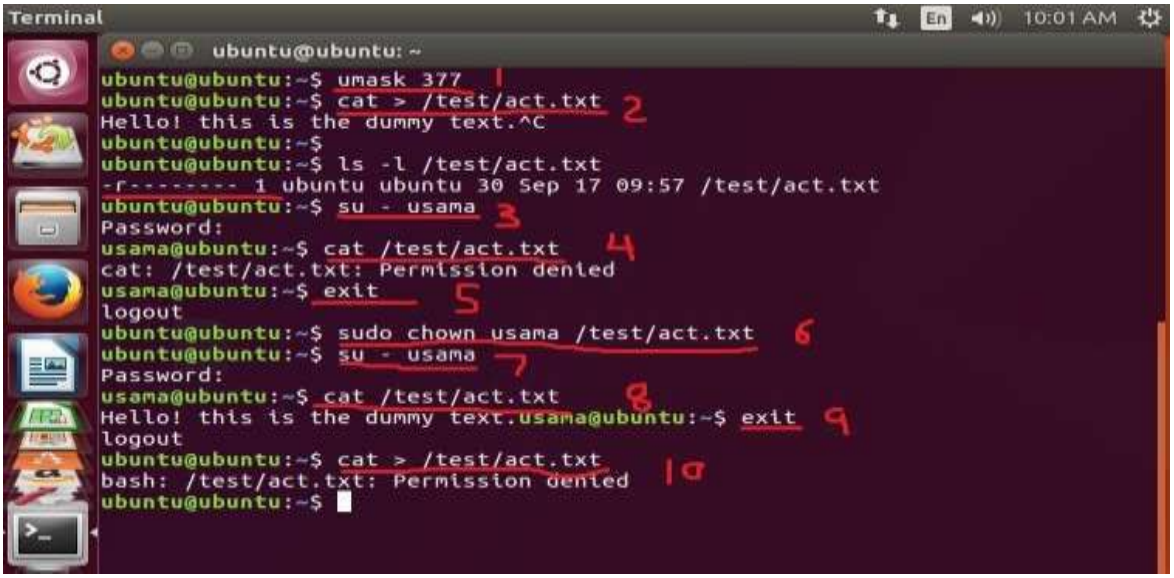


Activity 2:

Perform the following tasks

- ***Set the permission such that a newly created file is readable only to the owner***
- ***Create a text file "act.txt" in /test directory crated in previous activity***
- ***Run the shell as user "usama" (created previously)***
- ***Access the file "act.txt"***
- ***Logout as usama***
- ***Now change the ownership of the file from ubuntu to usama***
- ***Run the shell again as usama***
- ***Read the file "act.txt" using cat command***
- ***Logout as usama***
- ***Now access the act.txt with user ubuntu***

Solution:

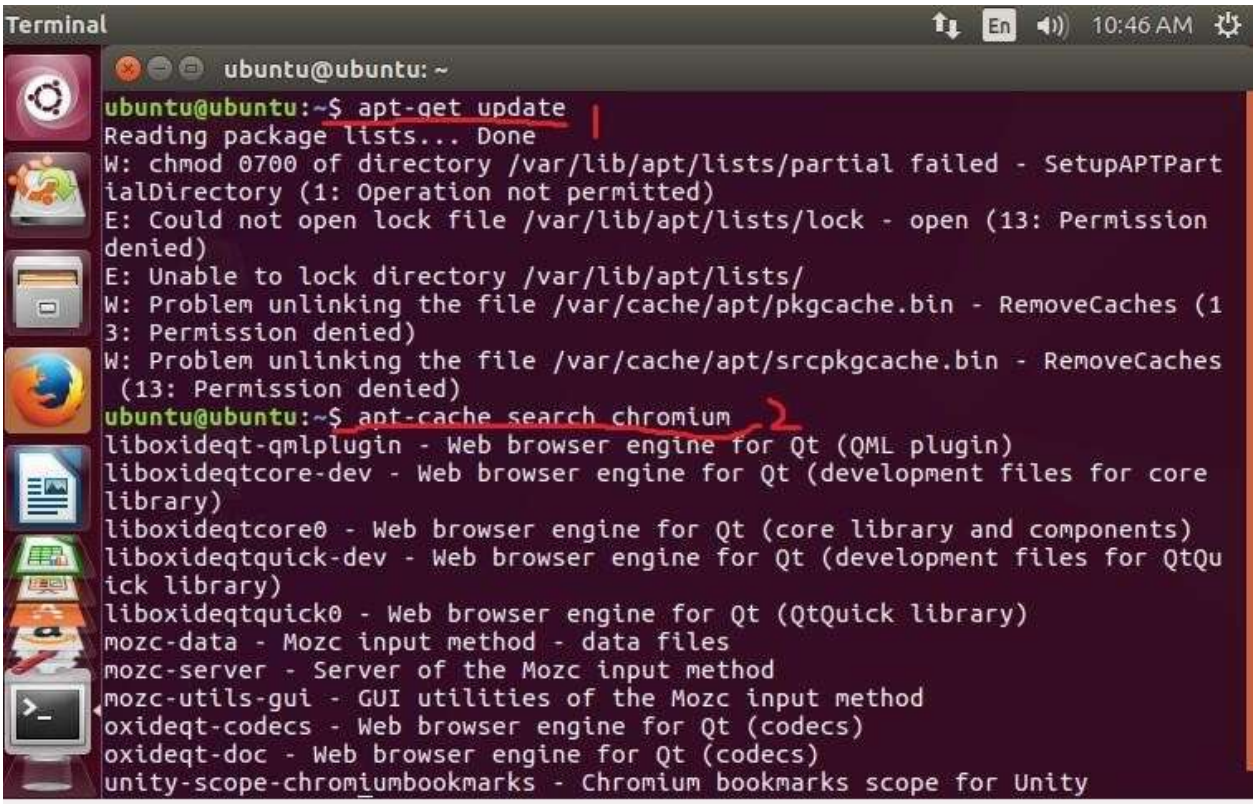


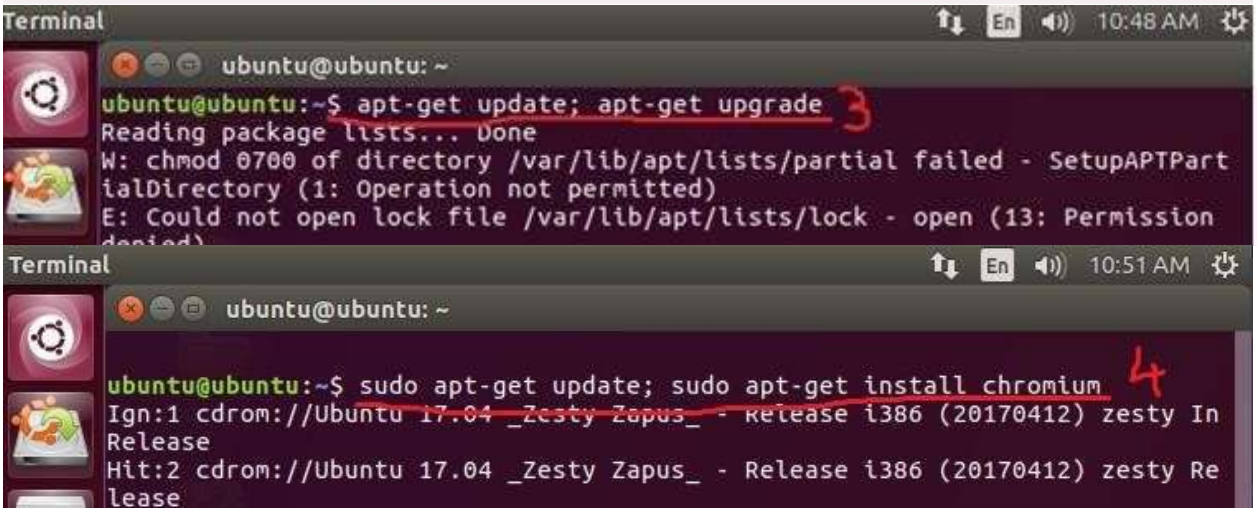
Activity 3:

Perform the following tasks

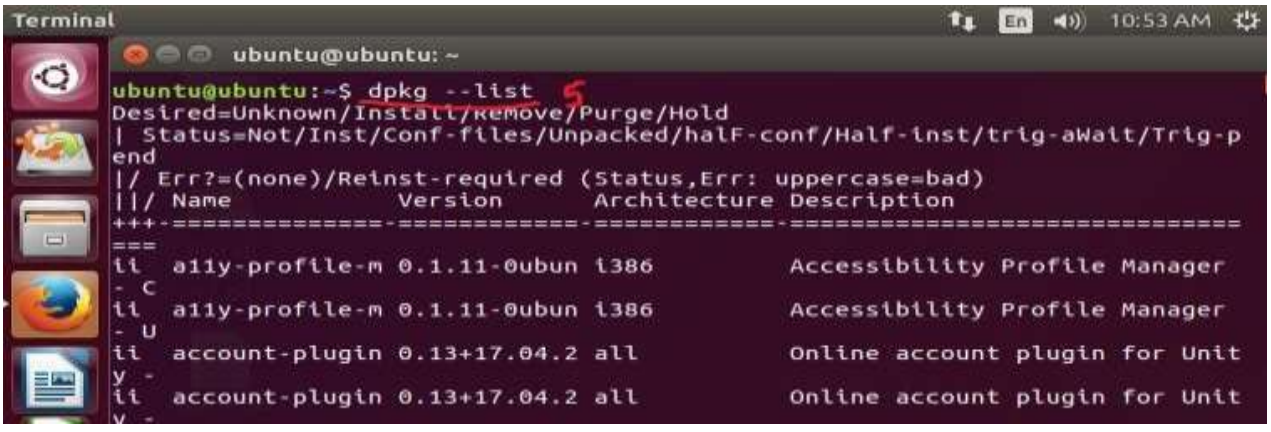
- ***search apt repository for the chromium browser***
- ***install the chromium browser using command line***
- ***write the command to update and upgrade the repository***
- ***list the software installed on your machine and write output on a file list.txt***
- ***read the list.txt file using cat command***

Solution:


Terminal
10:46 AM
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ apt-get update
Reading package lists... Done
W: chmod 0700 of directory /var/lib/apt/lists/partial failed - SetupAPTPart
ialDirectory (1: Operation not permitted)
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission
denied)
E: Unable to lock directory /var/lib/apt/lists/
W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (1
3: Permission denied)
W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCaches
(13: Permission denied)
ubuntu@ubuntu:~$ apt-cache search chromium
liboxideqt-qmlplugin - Web browser engine for Qt (QML plugin)
liboxideqtcore-dev - Web browser engine for Qt (development files for core
library)
liboxideqtcore0 - Web browser engine for Qt (core library and components)
liboxideqtquick-dev - Web browser engine for Qt (development files for QtQu
ick library)
liboxideqtquick0 - Web browser engine for Qt (QtQuick library)
mozc-data - Mozc input method - data files
mozc-server - Server of the Mozc input method
mozc-utils-gui - GUI utilities of the Mozc input method
oxideqt-codecs - Web browser engine for Qt (codecs)
oxideqt-doc - Web browser engine for Qt (codecs)
unity-scope-chromiumbookmarks - Chromium bookmarks scope for Unity


Terminal
10:48 AM
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ apt-get update; apt-get upgrade
Reading package lists... Done
W: chmod 0700 of directory /var/lib/apt/lists/partial failed - SetupAPTPart
ialDirectory (1: Operation not permitted)
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission

Terminal
10:51 AM
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ sudo apt-get update; sudo apt-get install chromium
Ign:1 cdrom://Ubuntu 17.04 _Zesty Zapus_ - Release i386 (20170412) zesty In
Release
Hit:2 cdrom://Ubuntu 17.04 _Zesty Zapus_ - Release i386 (20170412) zesty Re
lease


Terminal
10:53 AM
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ dpkg --list
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-p
end
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name Version Architecture Description
+++-==========-==========-==========-==========
ii a11y-profile-m 0.1.11-0ubun i386 Accessibility Profile Manager
- C
ii a11y-profile-m 0.1.11-0ubun i386 Accessibility Profile Manager
- U
ii account-plugin 0.13+17.04.2 all Online account plugin for Unit
y -
ii account-plugin 0.13+17.04.2 all Online account plugin for Unit

3) Graded Lab Tasks

Task 1:

1. Create three user accounts named user1, user2, and user3.
2. Create two groups named gr1 and gr2.
3. Add user1 to group gr1.
4. Add user2 and user3 to group gr2.
5. Check the User ID (UID) and Group ID (GID) by listing the file /etc/passwd.
6. Find the lines of the newly created users in /etc/passwd and identify the UID and GID for these accounts.
7. Write the command that shows the UID and GID of your current user.
8. Create three files using the touch command: file1, file2, and file3.
9. Set the following permissions using chmod with letters (symbolic notation):
    a. rwxrwxr-x for file1
    b. r-x--x--x for file2
    c. ----xrwx for file3
10. Prepare the following files/folder: file4, file5, and folder1.
11. Set the following permissions using chmod with numbers (numeric notation):
12. rwxrwxrwx for file4
13. -w------- for file5
14. rwx--x--x for folder1
15. Create two user accounts:
16. tst1 with login ID tst1, group users, shell bash, and home directory /home/tst1.
17. tst2 with login ID tst2, group public, shell bash, and home directory /home/tst2.
18. Set a password for both accounts.
19. Login as tst1 and copy /bin/ls into the home directory of tst1 as myls.
20. Change the owner of myls to tst1 and set the permission to 0710.
21. Explain what the permission value 0710 means.
22. Login as tst2 and try to execute /home/tst1/myls to list the current directory. Check whether it works or not.
23. Create a new group named labo and add tst1 and tst2 to this group.
24. Change the group owner of the file myls to labo.
25. Login again as tst2 and try to execute /home/tst1/myls to list the current directory. Check whether it works now or not.

Task 2:

Perform the following tasks:

1. Create a directory named labdata in your home directory.
2. Inside the labdata directory create two files named data1.txt and data2.txt.
3. Display the permissions of the files using the ls -l command.
4. Change the permission of data1.txt so that only the owner can read and write the file.
5. Change the permission of data2.txt so that everyone can read the file but only the owner can write.

6. Create a new user named student.
7. Switch to the student user using the su command.
8. Try to read data1.txt and data2.txt.
9. Logout from the student account and return to the original user.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation

LABORATORY SKILLS ASSESSMENT (Psychomotor)
Total Marks: 100

(Max Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S< 70%	Level 3 70% ≤ S< G0%	Level 4 G0%≤ S ≤100%	Score (S)
Procedural Awareness (20)	Selects inappropriate Linux commands, shell scripting techniques, or process management methods.	Selects and applies partially appropriate Linux commands and techniques	Selects and applies considerably appropriate Linux commands and techniques.	Selects and applies completely appropriate Linux commands and techniques	
Practical Implementation (20)	Makes major critical errors in executing Linux commands, scripting, and system processes.	Makes numerous critical errors in executing commands and process management.	Makes minor non-critical errors in executing Linux commands and system operations.	Executes Linux commands and manages processes correctly with no errors.	
Process Management and Shell Scripting (20)	Program logic contains major errors with incorrect or contradictory script flow.	Program logic has some errors with occasional contradictions in process execution.	Program logic is mostly correct but may contain occasional redundancy or minor errors.	Program logic is completely correct with no contradictions or redundant processes.	
Syntax Correctness and Results (20)	Program does not follow proper syntax for Linux commands and shell scripting, leading to incorrect outputs	Program partially follows proper syntax, producing correct results for few inputs.	Program adequately follows proper syntax, producing correct results for most inputs.	Program fully follows proper syntax, producing accurate results for all inputs.	
Use of OS Tools (10)	Uses OS tools (like terminal, process manager) with limited competence.	Uses OS tools with some competence.	Uses OS tools with considerable competence.	Uses OS tools proficiently with a high degree of competence.	
Safety (10)	Requires constant reminders to follow system safety procedures (e.g., file permissions, process handling).	Requires some reminders to follow system safety procedures.	Follows system safety procedures with minimal reminders.	Routinely follows system safety procedures.	
				Marks Obtained	

LABORATORY SKILLS ASSESSMENT (Affective)
Total Marks: 40

Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S < 70%	Level 3 70% ≤ S < G0%	Level 4 G0% ≤ S ≤ 100%	Score (S)
Attitude s Engagement (5)	Shows little interest in lab activities; does not participate actively.	Participates occasionally but lacks enthusiasm and consistency.	Engages actively in most lab activities with interest.	Highly motivated, participates enthusiastically, and shows a proactive approach	
Responsibility s Punctuality (5)	Frequently misses deadlines and is often late to lab sessions.	Occasionally late or misses deadlines but tries to catch up.	Submits work on time and attends lab sessions regularly.	Always punctual, meets deadlines, and takes full responsibility for assigned tasks.	
Collaboration s Teamwork (10)	Rarely collaborates, struggles to work in a team, and does not contribute effectively.	Works with team members occasionally but struggles with communication.	Cooperates well, contributes effectively, and maintains professional interactions.	Actively engages in teamwork, supports peers, and demonstrates excellent collaboration.	
Communicatio n s Presentation Skills (10)	Struggles to explain concepts, unclear verbal/written communication.	Communicates ideas with some clarity but lacks confidence or coherence.	Presents ideas effectively with minor issues in clarity or structure.	Communicates clearly, confidently, and effectively in all aspects of lab work.	
Report Quality (10)	Report contains many errors.	Report is somewhat organized with some spelling or grammatical errors.	Report is well organized and cohesive but contains some grammatical errors.	Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished.	
				Marks Obtained	