جامعة بيرزيت

BIRZEIT UNIVERSITY

Computer Science Department

Linux OS Laboratory Manual (V 1.2)

COMP311

Nael I. Qaraeen

Approved By:

Computer Science department

May 2015

Table of Contents

* pwd . tells you ~~who~~ where you are

Lab1. Introduction to Linux environment

Objectives

After completing this lab, the student should be able to:

- Log into and use a Linux system
- Learn the Linux command basic structure
- Use basic Linux commands to get familiar with the Linux environment
- Get help information on Linux commands
- Identify some important Linux Files

Logging in to Linux

Login directly to your Linux system by using your username (login name) and password (provided to you by the instructor). If you are using a client machine (machine running another Operating System such as Windows) then you need to use a remote access command such as telnet to login to the Linux server as follows:

Open the start menu from Windows
Type the following command in the search field:

telnet IP address of server (e.g. telnet 172.16.2.90)

This should provide you with a terminal where you can enter your login name and password to access the Linux system.
At the terminal prompt (e.g. [username@localhost ~]$) you can type your commands to interact with the system.

Linux Commands General Format

A Linux command may contain one or more of the following three categories specified in the following order:

Command *Option(s) Argument(s)*

a- Command name (such as ls, cp, or finger)

Type the command:
ls
Describe the result?
list directory contents (عرض محتويات الدليل)

b- Command options which indicate optional preferences on how you want the command to run. Those are usually specified using a – followed by a letter to

- ls is command - → {-a, -al} command option → - Argument etc

specify the option(s) (e.g. ls –al where the ls command has two options a (all) and l (long format)).

Type the command:

ls –a

Describe the result now? (all files)

show hidden files, do not ignore entries starting with .

Why did the files get listed in this case and not the previous case (hint: notice that all files names newly listed start with a dot (.))?

Because (.) it hides files and command ls-a displays hidden and unchidden files.

Type the command:

ls -al

Describe the result? use a long listing format

Show all files inside etc on the terminal.

c- Command Arguments which specify the parameters that you want your command to use while executing (e.g. ls –al /etc where /etc is an argument to the ls command specifying that you want to list the files in the /etc directory).

Type the command:

ls –al /etc } if we don't use it, it will take current directory

Describe the result?

Show all files inside ets on the terminal(directory).

(/) etc is root

rik c in windows

To be able to display the result one line (or one page at a time), you need to filter that result using filtering commands like ***more*** or ***less***.

Type the command: root file system gives data as pages

ls -al /etc | more

Press the space bar to get one page at a time.

Press Enter to get one line at a time.

Press letter 'q' to quit

An option (as the name specifies) is not needed to run a command, but arguments may sometimes be necessary to run certain commands (e.g. cp, rm, …)

Type the command:

cp

What happened?

missing file operant

Now type it with a source and destination file names (i.e. ***cp srcfile destfile***).

What happened? (Note: you can simply create a ***srcfile*** by running the following command before the ***cp*** command is run: ***touch srcfile***).

cp v1 v2] new file in same directory

cp v1 /etc/v2] etc directory

to connect to server (prefered)
command line interface ↑
windows desktop

whoami ⇒ your username

Getting Familiar with Linux Environment

Try the following commands to familiarize yourself with Linux:

1- *who* (displays information about who is logged in to the system at this time)
2- *finger username* (you may select any user name from the who command output)
What is displayed? How do you think the system knows all that information?

gives info on what that user is doing

3- *finger*
4- **w** (displays **what (w)** the users are doing on the system)

To be able to communicate with another user try the command:

5- *write username*
Type your message
Type ctrl-d to stop and exit

Such commands as write and talk may be distracting for users on the receiving end. Linux gives the user the ability to allow or deny such commands from disrupting his/her work. To do this a user uses the *mesg* command as follows:

mesg n (prevents others from sending messages using commands like write)
mesg y (allows other users to send messages)
mesg (displays whether messages are allowed (y-yes) or denied (n-no))

Help on Commands

There are several methods to get help on a Linux system. One of the most useful is the man (manual) pages.
To get detailed information about a command type:

man command name (e.g. man ls)

ls --help

This will provide you with information on the syntax of the command and a short description. It will also present the different options that may be used with that command. The *man* command uses the *less* command (similar to the more command) for displaying information. This means you can use the space-bar (one page at a time), Enter (one line at a time), and q (quit) to browse the given information.

The man pages are divided into sections. Each section gives information about the command in a different context as shown below:

Try the command:
man passwd

Gives information about ***passwd*** as a command (section 1 of the man pages) used to change your password. ***Read through it and use it to set a new password to your account.***

Did it work? _____.

Try the command:

man 5 passwd (5 is section 5 of the man pages)

This will give you information about ***passwd*** as a file on the system.

When you browse through the man pages of any command make sure you check out the see also section which provides you with the names of related commands to the one your exploring. The names of the commands are usually specified as:

command name(section number) (e.g. ***passwd(5)*** – meaning ***passwd*** in section 5 of the man pages).

Using the man pages do the following:

1- Find what the command du is used for with the options -h and -s;

du command: Summarize disk usage.

-h: human readable. (size)

-s: summarization. (summary of size)

2- What is the clear command used for? Try it.

Clear the terminal screen.

3- List any two new commands that you haven't seen before and find out what they do. Hint: check the directories /usr/bin and /usr/sbin.

Command 1: chfn.

Function of command:

Change rear user name and information.

Command 2: chat.

Function of command:

Automated conversational script with a modem.

Important Linux Files

There are two important Linux files that contain information regarding user and group data. The first is **/etc/passwd** and the second is **/etc/group**.

cd ~ : takes you back to home directory

/etc/passwd file:

Type the command:

more /etc/passwd list all users that have login

(hit the space bar until you see a line that starts with your user name)

The line will be similar to something like:

u1112233:x:520:66:Ahmad_Hamdan:/home/students/comp311/u1112233:/bin/bash

use the man pages (section 5 for passwd to see what the fields in this line represent).

u1112233: Username for log in.

x: Encrypted password.

520: user id ID.

66: Group ID.

Ahmad_Hamdan: Real username.

/home/students/comp311/u1112233: user space.

/bin/bash: default use shell.

Where do you think the finger command gets the information it displays about users?

/etc/ passwd.

/etc/group file:

Type the command:

more /etc/group

You will find lines similar to the following:

students:x:66:ahmad,u123456

Use the man pages (section 5 for group to see what the fields in this line represent)

students: Group name.

x: Encrypted password.

66: Group number.

Ahmad,u123456: Group user.

Lab2. Text editing (vi editor)

Objectives

After completing this lab, the student should be able to:

- Become familiar with the visual (vi) editor modes
- Use the vi editor to manipulate simple files
- Write and use simple vi macros

Linux Editors

There are several text editors on Linux but one of the most commonly used is the ***vi*** (visual) editor which was recently upgraded to ***vim*** (vi modified). Using the ***vi*** editor may seem strange at the beginning, but once you get used to it you will rarely use any of the other editors. This editor is a default on most (if not all) Linux and UNIX based systems. It provides the user with very powerful editing capabilities using very simple tools (a keyboard and a simple terminal).

Vi Modes

To get started with ***vi***, you need to understand that when using this editor you are usually in one of two modes:

1- Insert (input) mode: which is used to type characters in a file.
2- Command mode: which is used to run commands (edit) file content

When you first start a vi editing session, you are in the command mode by default. To switch to insert mode you use one of the following letters:

i-	Insert before current character
I-	Insert at the beginning of current line
a-	Append after current character.
A-	Append at the end of current line
o-	Open new line below current line.
O-	Open new line above current line.

To switch back to the command mode you can use the ESC (Escape) key.

Editing A Simple File

Let us go ahead and start editing a simple file.

Type the command:

vi myfirst

Now press the letter ***i*** to change to insert mode and type the following (on three separate lines):

Your full name
Your student id number
Your major

To finish your vi session, you need to switch back to command mode (*Press ESC*).
Type a colon (:). This will move your curser to the bottom left side of your page. This is called the ex mode (which is an extension to the command mode). To quit your file type one of the following:

q! - to quit without saving recent changes
w - to save recent changes and not quit
wq - to save recent changes and quit session

Save and quit the file and then open it again using the command: ***vi myfirst***

To move around in a vi session you can use the following letters (in command mode):

h - move left j- move down k - move up l (el) - move right

To delete any mistakes you make in a file use one of the following (in command mode):

x – delete current character
#x – delete # of characters (e.g. 4x – deletes 4 characters starting with current)
dw - delete current word
#dw – delete # of words (e.g. 3dw - deletes 3 words starting with current word)
d$ (or dd) – delete current line
#dd- delete # of lines (e.g. 5dd – deletes 5 lines starting with current line)

Try deleting your last name, the last number of your student id and your major line from the file myfirst.

The content deleted in the above commands is actually saved in the vi buffer (like the cut command). To be able to restore that content back you may use the commands:

p – paste left or below
P - paste right or above

If you want to copy/paste instead of cut/paste then use a y (yank) instead of a d(delete) as follows:

yw – yank word
#yw – yank # words
y$ (or yy) – yank line
yy – yank # lines

To modify your file content use one of the following (in command mode):

r- replace current character (e.g. re changes the current character to e)
cw - change word (e.g. cwnew changes current word to new)
#cw – used to change # words with new words.
c$ - change line
#c$- change # of lines with new lines.

If you decide to undo any of the changes you made to your file, use one of the following:

u – undo last change
U – undo all changes in current line
:e! – undo all changes since last save

To search a file forward for a certain pattern, you use the / (slash) followed by the pattern. For example to search for the word hello in a file you type: */hello*. To get the next position of the pattern type ***n*** (next).
To search a file backwards use a *?* instead of a /.

Vi Macros

You can create and use macros in *vi*.
To create a macro in command mode use the map command in the ex mode as follows:

:map S 1GddGp:wq

which will create a command mode macro called ***S*** which when pressed will go to the first line (1G) cut the line (dd) move to the end of file (G) and paste the line after (p) and then will save file (:w) and quit (q).

To create a macro in input mode you use the ***map!*** command rather than map. To include control characters such as ESC or ENTER in your macro, you do the following:
Press the ctrl key then v key then the char you want. i.e. to include the ESC key in my macro I press Ctrl then v then ESC and it will be saved as the control char ^] .
To undo a macro, you use the ***unmap*** command followed by the macro's name (e.g. : ***unmap S***).

Macros will disappear after you exit your vi session. To make your macros permanent, you need to include them in a file called ***.exrc*** or ***.vimrc*** under your home directory.

Create the following macros and put them in file .exrc under your home directory:

Macro 1: called H which is an input mode macro that copies the last three lines of the file to the beginning of the file. :map! H^[Gy2kGP:wq^M

Macro 2: called S which is a command mode macro that replaces the current word with the word new. :map S cw:wq^[

Save the .exrc file and then do the following:

Create a new file called testmacros and fill it up with any five lines.

open the file testmacros and apply your macros (H and S) to it. Did they work?_____.

More vi Practice

In the brackets, write the command(s) that you would use to do each of the following:

1. **Type vi linux**
2. **Change to input mode (i,I,a,A,o,O),o**
3. **Type the following text**

Linux was first created by Linus Torvalds
from Finland.
He based it on the UNIX OS
which was first created by Ken Thompson.
Linux is open source.
There are different Linux distributions,
such as RedHat
and Ubuntu.
Linux is a very stable and popular Operating System.
Using Linux is fun.

Go back to command mode (ESC).

4. **Go to the beginning of the first line in the file linux (gg) , 1G , :1 , 1GB**
5. **Go to the beginning of the last line (shift+g , G , :G)**
6. **Go to the ninth line (9G)**
7. **Use the letters h, j, k, l, and w to move the cursor to the beginning of the word "Red" and then delete three chars. (3x)**
8. **Insert the word "Black" (i)**
9. **Search for word open and move there (?open).**
10. **Delete the word "open" by using a single command (dw)**
11. **Undo this last deletion (u)**
12. **Using the search facility move your cursor to the sentence "Linux is a very stable ...".**

(/linuxexpression).

13. **Using one command, change the word "is" to "was" (cw)**
14. **Move back to the first line of the file (1G)**
15. **Copy four lines into the buffer (4yy , :4y)**
16. **Move to the end of the file (G) and put the copied lines there (p)**
17. **Write a command mode macro called B that will save the file and quit.**

(:map! B^[:wq!^M)

18. **Write (save) the file and quit using the macro in 17.**

Lab3. File Systems (I) (Structure and File Types)

Objectives

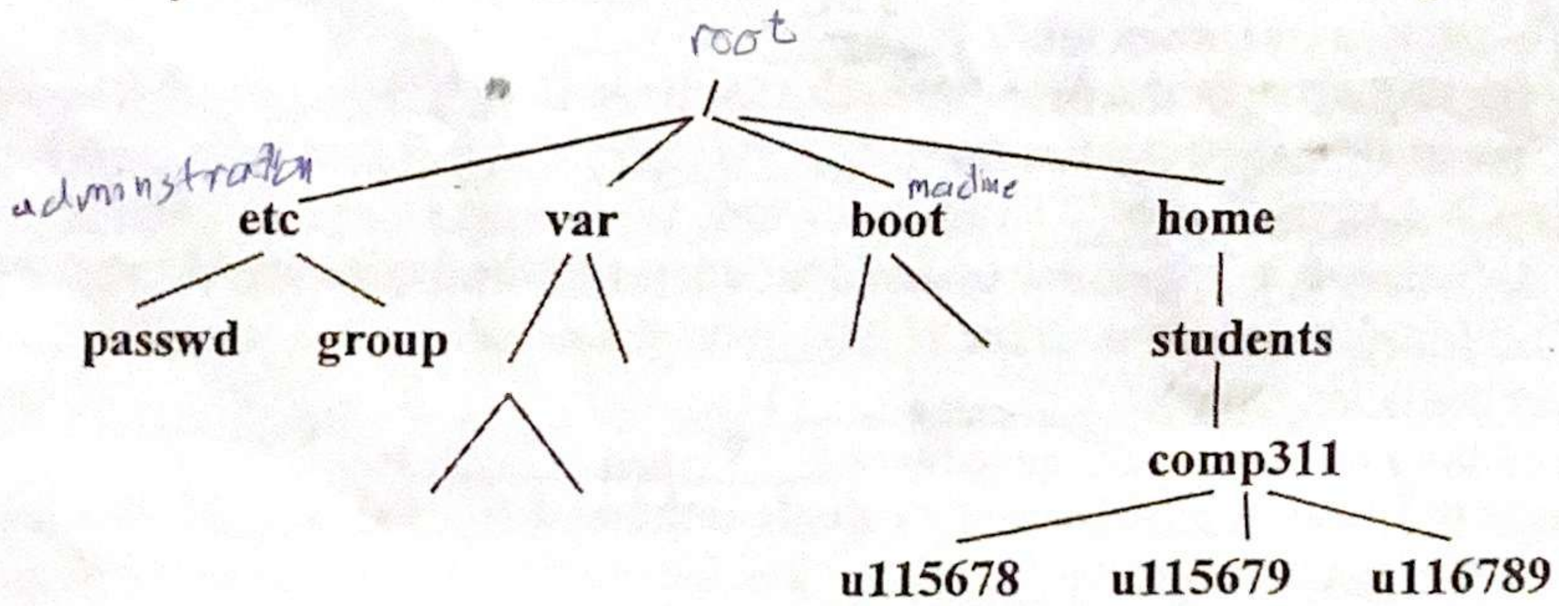
After completing this lab, the student should be able to:

- Understand the structure of the Linux file system
- Build tree structures using absolute and relative paths
- Recognize and create the different main Linux file types

Linux File Systems

A File System is a data structure that contains data about files and how they are organized. It is a logical name that represents a physical device such as a partition on disk.

The File Systems on Linux usually have directory names such as /home or /usr/local.
Linux must at least have one file system which is the root file system (*/*).
Linux file systems are hierarchical tree structures similar to the following:



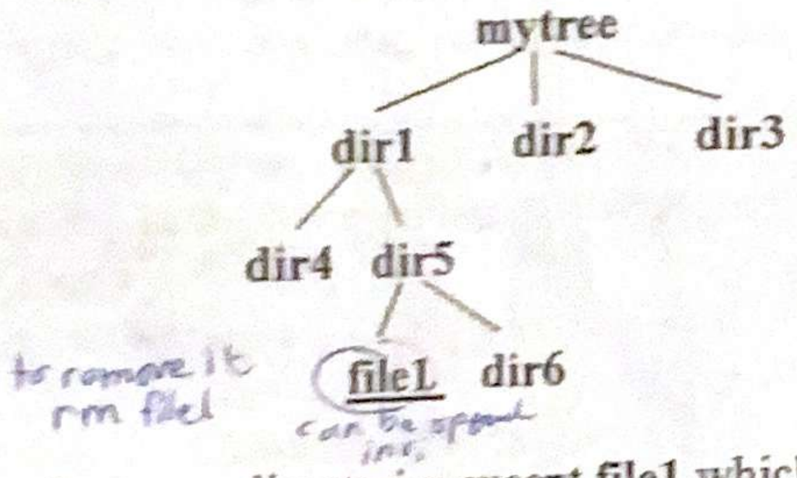
There are several common directories that usually exist in most Linux systems such as:
/etc: includes most administration related files.
/var: includes data that changes such as log files and user mail boxes.
/boot: includes system boot files.
/home: includes user home directories

To display the file systems, use the command *df* (display file systems) as follows:

df -h (-h human readable format).

What are the physical device names as well as the mount points (directory names where file systems are mounted) for the file systems on your system?

To Manipulate directories under a file system, you can use the following commands:
mkdir newdir (creates a new directory called newdir)
cd newdir (changes your position to newdir)
rmdir newdir (removes directory new directory only if newdir is empty)
rm –rf newdir (removes non-empty or empty directory newdir)
pwd (displays present working directory)
cd → takes you to home directory
history →
Using the commands above, create the following tree under your home directory:



All the above are directories except file1 which is a regular file.

Commands used to create *mytree* in order:

```
mkdir mytree
mkdir dir1 dir2 dir3
cd mytree
mkdir dir1 dir2 dir3
cd dir1
mkdir dir4 dir5
cd dir5
touch file1
mkdir dir6
```

To display your tree use the command:
ls –R mytree you need to be in the higher level to see it.

Absolute and Relative Paths

Any Linux file may be referenced by either its absolute path name or relative path name. Each file has one and only one absolute path name while it may have an infinite number of relative names.

The absolute path name for file1 in the previous tree is:
/home/students/comp311/username/mytree/dir1/dir5/file1

While it has several relative names such as:
./mytree/dir1/dir5/file1
./mytree/../mytree/dir1/dir4/../dir5/file1

and so forth. Notice that (.) stands for current directory while (..) stands for previous (parent) directory.

Remove the sub tree *mytree* created in the previous section (***rm –rf mytree***)
Now try creating the whole tree again using relative paths from your home directory (i.e. you are not allowed to use the *cd* command to create any parts of the tree).

Commands used:

```
mkdir mytree
mkdir mytree/dir1
mkdir mytree/dir2
mkdir mytree/dir3

mkdir mytree/dir1/dir4
mkdir mytree/dir1/dir5
mkdir mytree/dir1/dir5/dir6
touch mytree/dir1/dir5/dir6/file1
```

relative path
`mkdir -p mytree/dir1/../dir2/../dir3/../dir1/dir4/../dir5/dir3`

So far, we have mentioned two types of Linux files as follows:

1- Regular files which include scripts, binaries, as well as text files. These files contain data and are identified by any empty first slot when we list them using the ***ls –al filename*** command. These are created using the ***vi*** editor.
2- Directories which are simply containers that include the mappings between filenames and subdirectory names and their unique inode (index) number in the file system. These are identified by the letter *d* in the first slot when we list them using the ***ls –al dirname*** command. These are created using the ***mkdir*** command.

The third type of Linux files are the special (device) files which are usually located under the */dev* directory. There are two types of device files:

1- Character device files: which are used to read and write from/to devices one character at a time (e.g. keyboard device files). These are identified with the character *c* when we list them with the ls –al command.
2- Block device files: which are used to read and write from/to devices one block at a time (e.g. disk device files).). These are identified with the character *b* when we list them with the ls –al command.

Run the command ls –al on the /dev directory. List three character device files and three block device files.

tab → command

./../File1

crw-rw-rw character device file

~~brw-rw---- block block device~~

brw-rw---- loop3 block device

srw------- mceloy

The fourth type of Linux files are the links.
The original links in Linux are called the hard links and are created using the command *ln*.
Create a directory to try some links (***mkdir links; cd links***)
Create a file called original and put the phrase "this is original" inside then save and quit (***vi original***)
Now create a directory called mydir (***mkdir mydir***)

Let us now start working with links:

Create a hard link called filehlink to file original:

ln original filehlink

List the files in your directory with ***ls –ali*** (the i option displays the inode numbers)
Notice that all the properties of file original and link filehlink (except the name) are exactly the same (even the inode number). Hard links basically give a new name to the same inode number.
Hard links have two limitations:

1- Not allowed on directories
   Try the command:
   ln mydir dirhlink
   What was the result?
   hard link not allowed for directory.

2- Not allowed across different devices (file systems)
   Try the command:
   ln /etc/passwd passwdhlink
   What was the result?
   Invalid cross-device link

Those limitations are solved using a different type of links called symbolic (soft) links.
To create a symbolic link simply use the option (-s) with the ***ln*** command.

Try the following commands and see what happens:

rm filehlink
ln -s original fileslink
soft

note
ls /mytree: from root
ls mytree
ls ./mytree: from current position.

*more: بتعرض محتوى

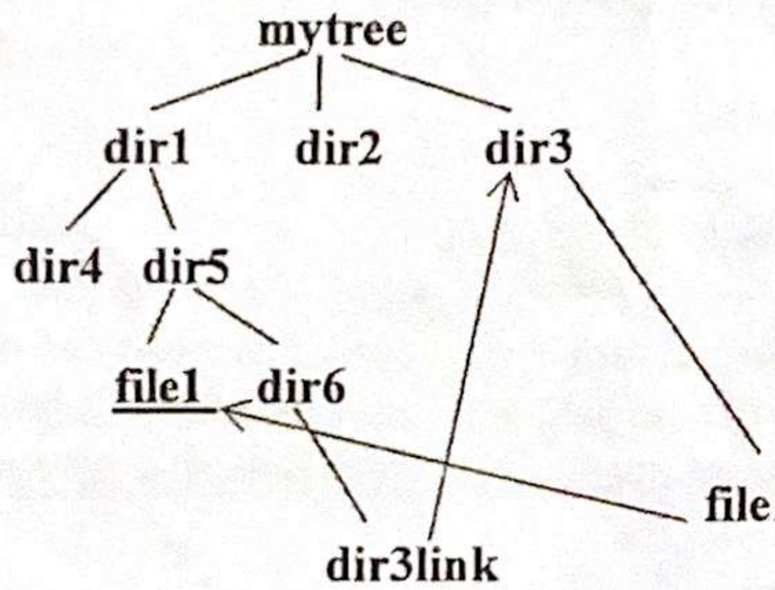
ls –ali (what are the differences between original and fileslink properties?)

Id's are different.

ln –s mydir dirslink (what happened now?)

ln –s /etc/passwd passwdslink (what happened now?)

Now go back to the tree you created earlier (***mytree***) and add the links shown below:



What commands did you use:

1. cd mytree/dir3
   ln ./../dir1/dir5/file1 file1link
2. cd mytree/dir1/dir5/dir6
   ln -s ./../../../dir3 dir3link

Although there are few more types of Linux files such as pipe files or socket files. They are rarely used or seen by users.

Another solution : ln -srf mytree/dir1/dir5/file1 mytree/dir3

Lab 4. File Systems (II) (File Metadata)

Objectives

After completing this lab, the student should be able to:

- Understand and manipulate permissions (mode) on different Linux files
- Set the default permissions for files and directories
- Identify and handle file properties such as ownership, groups, size, and timestamps.

Permissions (Mode)

Each file has nine characters that represent the permissions on that file. Those are divided into three equal parts:

user (u) = user (owner) permissions on the file.

group (g) = permissions of members of the group name stamped on the file (except owner).

other (o) = all system users other than the group and owner.

The main three permissions that may exist on the file are ***read (r)***, ***write (w)***, and ***execute (x)***. These mean different things for files than they do for directories as follows:

Read: for files it means the user can view content of file (using vi, more, cat, …) while for directories it means the user can view content of directory (using ls).

Write: for files it means the user can modify the content of the file, but is not allowed to remove it while for directories it means the user can modify the content of the directory (i.e. can create or remove files and subdirectories).

Execute: for files it means the user can run the file (scripts or binaries) while for directories it means the user can access the directory (use cd).

To change the mode, a user may use the ***chmod*** (change mode) command. This command can specify the new permissions using a relative or absolute method.

Chmod using relative method

Using this method the user can modify the permissions on a file (or directory) relative to the already existing permission as follows:

Assume that we start with the following permissions on a file called myfile *r_xrw_r___*

The command: ***chmod u+w,g-rw,o+x myfile***
Will change the permissions on myfile to *rwx___r_x*
If we continue with the command: ***chmod u=rw,g+w myfile***
The permissions will now become *rw___w_r_x*

Check the man pages on command ***chmod*** for more examples and then do the following:

rwx
1 → 000
2 → 001
3 → 010
011
:
7 → 111

Create a directory called ***mode*** and move inside it (***mkdir mode; cd mode***).
Create a file called ***myfile*** and a directory called ***mydir*** inside directory ***mode***.
Using the ***chmod*** command with relative mode, change the permissions on both ***myfile*** and mydir as follows:

rwxr_xrw_ ***commands=*** chmod u=rwx, g=r-x ;_____.

r__rw____x ***commands=*** _____;_____.

___rwx_wx ***commands=*** _____;_____.

The second method is the absolute method which does not depend on the permissions that already exist on the file.

This method uses a binary 1 where you want a permission to be set and a binary 0 where you want it unset as follows:

The command: ***chmod 734 file*** will set the permissions on ***file*** to 111 (7) for user = rwx and 011(3) for group = _wx and 100(4) for other = r__ so the permissions on the file will be= rwx_wxr__

Using the ***chmod*** command with absolute mode, change the permissions on both ***myfile*** and ***mydir*** as follows:

rwxr_xrw_ ***commands=*** chmod 756 myfile ;_____.

r__rw____x ***commands=*** chmod 461 myfile ;_____.

___rwx_wx ***commands=*** chmod 073 myfile ;_____.

D = Directory = 777
f = file = 666

← Default Mode

The default permissions that are set on newly created files and directories are set using the ***umask*** command.

Run the command:

umask

What number did you get: 0022 .

This number decides the permissions set on newly created files or directories.
Read the manual page for umask (***man 2 umask***) and try to figure out what permissions would you get on files and directories after you run the command: ***umask 123***

Expected permissions on a new file= 666 .
Expected permissions on a new directory= 777 .

softlink: بعمل زي ما هو الفايل

hardlink: بوجه بتغيير الفايل

ls -ali → بتورجيني رقم الفايل

Check to see if you understood how ***umask*** works by creating a new file and a new directory and checking the set permissions. Did you get it right? (☺)_____.

What permissions would you expect after the command: ***umask 625*** is executed. Try it to see the results. Did it work? (☺)_____.

Now let us try and do the reverse:

If you want a newly created directory to have the permissions ***rwxr___wx what umask command would you run:*** 743 umask 034 .
Try it. Did it work? Yes .
To have the following permissions on a newly created file: r__rw___w_ what umask command would you run: 462 unmask 204 .
Try it. Did it work? Yes .
What about if you wanted a newly created file to have permissions: rwxr___wx. What umask command would you run: 741 umask 036 .
Try it. Did it work? No . ***Why?*** File takes off X (for security)

Changing Link Properties

Since we can modify the mode property of a file we can do more testing to see how links work. Go back and create two files called ***file1*** and ***file2*** and then create a hard link called ***hlink*** to ***file1*** and a symbolic link called ***slink*** to ***file2***. List the commands you used:

```
touch file1 file2
ln file1 hlink
ln -s file2 slink
```

Now try changing the permissions on file1 to rwxrwxr__.
Command: chmod 774 file1

What happened to the permissions on hlink? Why?

Became like file1 .

Now change the permissions on hlink to rwx_____x.
chmod 701 hlink

stat — ع طلع directory's

*others: are the all things that are not groups or user

Command: chmod 701 hlink.

What happened to the permissions on file1? Why?

it changes to file1.

Now try changing the permissions on file2 to rw_r_xr__.

Command: chmod 654 file 2

What happened to the permissions on slink? Why?

didn't change.

Now change the permissions on slink to r__rwxr_x.

Command: chmod 475 slink.

What happened to the permissions on file2? Why?

didn't it change.

What happened to the permissions on slink? didn't change.

Ownership and Groups

The next file property is the name of the owner of the file. The owner is the only user (other than root) that can modify the properties of a file. The root is the only one that can change a file ownership using the command ***chown*** as follows:

chown newuser filename

Try changing the ownership of any of your files. Did it work? No.

The following file property is the group name on the file. This group name may be modified by the owner if he/she is a member of the new group he/she wants to put on the file. To change the group, a user uses the command ***chgrp*** as follows:

chgrp newgroup file

Try to change a group on any of your files. What happened?__________________.

Size

The next property shows the size (in bytes) of a file. Try creating a file and putting the phrase "how are you" inside then save and quit. **What is the size of the file?** 12 byte.

Why? 1char= 8bit (ASCII) = 1byte , 11char+1 char end of file

Change to directory /dev. Command: cd /dev.

Check out the size property on device files. What did you find?

ls -l sda*.

...after executing the last command,

brw-rw---- 1 root disk 8, 1 Feb 8 2008 sda1

What are the two numbers that exist instead of the size?

devices: Minor id: specific devices type (0..7)

class: Major id: categorize devices(8)

Go back to your home directory. Command: cd, cd~

Go back and display the size of the symbolic link (slink) you created earlier. Can you figure out how that size was calculated? cd mode4s -al, 5byte

Try creating a new symbolic link and see if you are able to figure out how the size on a symbolic link is set. What did you find?

The length of file name (# of characters)

✱ Time Stamps

A file has several time stamps. The main two are:

1- Last modification time: which is the time the file was last modified and saved.
This is the default time displayed by ls –al command.

2- Last access time: which is the time the file was last accessed or viewed. What ls option is used to display that time. ls -lu (Check the man pages).

Check the times on file ***myfile*** and record them.

Now view the file using the ***more*** command. ***What happened to the times?***

17:01 shows the data without accessing it, changed. didn't change

Now open the file ***myfile***, modify it and then save and quit.

What happened to the times now?

changed time

Another way to display file properties in detail is to use the ***stat*** command. Run the ***stat*** command on file ***myfile*** as follows:

stat myfile

What information can you see:

File, Size, Blocks, IO Blocks, links, Inode, Uid
Device, Access, Modify, change, birth, Gid

For more information on the output, you can read the man pages on the ***stat***.

File name

$2^8 - 1$

A Linux file name can be up to 255 characters long and is made of any characters. A dot has no special meaning in a file name except if it is the first character then the file is a hidden file. ***Create a hidden file called .hidden.***

Command: touch .hidden.

Try to list your files using the command ls. Can you see .hidden?

No.

Now try to list the files using the command ls with the –a (all) option? Can you see it now? yes.

Lab5. Shell Usage and Configuration (I)

Objectives

After completing this lab, the student should be able to:

- Become familiar with common Linux shells.
- Recognize and manipulate system and user defined shell variables.
- Identify and use shell functions like command substitution, aliasing, command line editing and file name completion.

Linux Shells

A shell is a command interpreter. It is the main program used by the user to access and use the Linux operating system. When the user enters a command and hits the Return key the shell checks the command and rejects or accepts it. Accepted commands are then passed on to the Kernel part of the OS for execution and the result is displayed by the shell to the user.

There are different shells on many Linux and UNIX based systems such as:

sh (bourne shell)
csh (C shell)
ksh (korn shell)
bash (bourne again shell)

To change from one shell to another simply type the name of the shell on the command line and hit Enter.

Shells have many features and functions which allow them to perform their work. The following are some of those features or functionalities:

Variable Substitution

A shell is a program that has several variables that help the shell do its work. Many of those variables are system defined variables (usually written using upper case letters) and some may be user defined variables.

Let us consider some of the system defined variables to see how the shell uses them.

PATH variable:

Run the command:

echo PATH

What did you get? _____.

Now run the command:

echo $PATH

What did you get? _____.

To display the value of a variable you need to precede it with the ($) character.
The **PATH** variable is used by the shell to locate commands for execution. Let us see how the shell is affected by modifying that variable.
Run the following commands:

SAVEPATH=$PATH (saves the value of PATH in variable SAVEPATH)
ls → *Did it work?* Yes
PATH=/etc
ls → *Does it work now?* no.
Why? The path for the command ls (/bin) was changed, so the interpreter does not identified
Restore the original value for variable PATH.
Command? PATH = $SAVEPATH

Now try the command:
ls → *Does it work now?* Yes.
You can add directories to your PATH. To add the dir /etc to the end of the PATH use the command:
PATH=$PATH:/etc
Try it and then use the command:
echo $PATH
Was it added as expected? Yes.

PWD and PS1 Variables:

Display the value of the PWD variable.
Command? echo $PWD (Print the current directory working)
Change your directory to /etc. Command? cd /etc.
What is the value of PWD now? echo $PWD or pwd ⇒ /etc
How do you think the pwd command works? cd\ cd\ cd\ ; ech $PWD.

Now run the following command:
PS1="hello >"
What happened to your prompt? Hello>.
Now run the command:
PS1='$PWD >'
What happened? /home/students/sec3/u1201112/Lab-5>.

There are several other variables such as HOME, PS2, SHELL, MAIL and so forth. To display the variables in your shell run the command:
set | more
List three more variables other than the ones mentioned above and their values:

1- Bash.
2- TTY.