PRINCESS NORA BINT ABDULRAHMAN UNIVERSITY

College of Computer and Information Sciences
Computer Operating Systems (NET 321)
1432-1433 H
Lab 5

The topics:

- **Creating files and directories**
- **Create empty files**
- **Moving, copying, and deleting files**
- **Using file-matching metacharacters**
- **Using file-redirection metacharacters**

Creating files and directories:

$ mkdir directoryName

Short for make directory this command is used to create a new directory.
Example:

1. Go to your home directory: Type ***cd.***
2. Make sure that you got to your home directory: Type ***pwd***
3. Create a new directory called **test** in your home directory***: mkdir test***
4. Check the permissions of the directory by typing: ***ls –ld test***
5. Suppose that you want to prevent everyone else from using or viewing the files in this directory: ***chmod 700 test***
6. Make the test directory your current directory : ***cd test***

NOTE:

$ ls –ld test	**provide a long listing of the test directory, without showing the contents of the test directory.**

The -d option tells ls not to list the contents of the test directory; just show us the listing for the directory itself.

To create a directory(test) in the Desktop, we have two ways:

- mkdir /home/userName/Desktop/test
- cd /home/userName/Desktop
  then : mkdir test

Create empty files:

$ touch *Filename*

When used without any options, touch creates new files for any file names that are provided as arguments

ex.:

The following command would create three new, empty files named *file1*, *file2* and *file3*:

```
$ touch file1 file2 file3
```

Moving files:

- **The command mv moves or renames files.**

To rename file:

$ mv *oldfilename newfilename*
$ mv abc1 abc2 → It will rename the file abc1 to a new name abc2

To move a file to a directory. use (mv) in the following form:

$mv *filename directoryname*
It will move the file into the named directory keeping its old name.
$ mv abc2 /home/ubuntu/test → Move abc2 file to test directory

Copying files:

Is similar to moving the files but you use *cp* instead of *mv*
$ cp *filename directoryname*

Deleting files:

To remove a file from the current directory
$ rm *filename*
$ rm * → * remove all files in the current directory

Using file-matching metacharacters:

To be able to refer easily to a group of files, the bash shell lets you use metacharacters. Anytime you need to refer to a file or directory, such as to list it, open it, or remove it, you can use metacharacters to match the files you want. Here are some useful metacharacters for matching filenames.

Characters	Meaning
*	This matches any number of characters(zero or more characters).
?	This matches any one(single) character
[…]	This matches any one of the characters between the brackets, which can include a dash-separated rang of letters or numbers.

The next few commands show you how to use shell metacharacters to match filenames so they can be used as arguments to the **`ls`** command. Using metacharacters shown below, you can match the filenames you just created with the `touch` command ***:***

```
$ touch apple banana grape grapefruit watermelon
```

1. ***This matches any number of characters*** *******

```
$ ls  a*
apple
$ ls  g*
grape   grapefruit
$ ls  g*t
grapefruit
$ ls  *e*
apple  grape  grapefruit  watermelon
$ ls  *n*
banana  watermelon
```

2. ***This matches any one(single) character*** ***?***

```
$ ls  ????e
Apple   grape
$ ls  g???e*
grape   grapefruit
```

3. ***This matches any one of the characters between the brackets […]***

$ ls [abw]*
apple banana watermelon
$ ls [agw] * [ne]
apple grape watermelon
$ ls [a-g] *
apple banana grape grapefruit

Using file-redirection metacharacters:

you can use less than (<) and greater than (>) signs to direct data to and from files. Here are the file redirection characters:

- < — Direct the contents of a file to the command.
- > — Direct the output of a command to a file, **overwriting** any existing file
- >> — Direct the output of a command to a file, **adding** the output to the end of existing file

Here are some examples of command lines where information is directed to and from files:

$ mail root < ~/.bashrc

$ echo "I finished the project on $(date)" > ~/*your fileName*

$ echo "I finished the project on $(date)" >> ~/*your fileName*

In the first example, the contents of the .bashrc file in the home directory are sent in a mail message to the computer's root user.

The second example, direct the output of a command" *I finished the project on Tues Oct 18 13:46:49 PST 2011*" ***to yourfile, overwriting any existing file*** .

The final command , , ***adding the output*** " *I finished the project on Tues Oct 18 13:46:49 PST 2011*" ***to the end of existing file.***

Helpful sites on lab5:

http://book.chinaunix.net/special/ebook/RedHat_Linux_Bible/8140final/LiB0044.html
http://docstore.mik.ua/orelly/linux/lnut/ch07_03.htm
http://www.howtogeek.com/howto/29980/whats-the-difference-between-single-and-double-quotes-in-the-bash-shell/
http://linuxcommand.org/wss0060.php
http://www.linux-tutorial.info/modules.php?name=MContent&pageid=20