Lab6. Shell Usage and Configuration (II)

Objectives

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

- Understand and use shell input, output, and error redirection.
- Use pipes to join several Linux commands into single powerful commands.

I/O Redirection

Commands (and programs) usually receive input and then produce output and error. By default the input is usually received from the keyboard and the output and error are usually both directed to the screen. Linux shells allow us to change those defaults and redirect input, output, and errors.

Input Redirection

Create a file called ***message*** and type the following two lines inside:

This is my message file
Goodbye

Then save and quit

The ***tr*** (translate) command is a useful command used to change input characters and may be used to encrypt characters.

Run the command

```
tr  "a-z"  "A-Z"
how are you
```

The result is "HOW ARE YOU". As you can see the input was received from the keyboard. You may redirect the input to come from the file message you created earlier as follows using the redirection operator (<):

```
tr  "a-z"  "A-Z"  <  message
```

What was the output?

You can append the redirected input using the here text (<<). Run the following command:

```
tr  "a-z"  "A-Z"  << !
> hello
> how are you
> hope well
> bye
> !
>
```

What did you get as output?

Output Redirection

The output of commands is sent to the screen by default. You may redirect the output by using the (>) character. Run the command:

```
ls –al
```

The output will be shown on the screen.
Now run the command:

```
ls –al > lsfile
```

No output will be displayed on the screen. View the file **lsfile** using the ***more*** command. It should contain the output of the "***ls -al***" command.

Using the (>) character will create a new file or overwrite an existing file.
To append the output to a file, you can use the (>>) character as follows:

```
ls  -al >> lsfile
who  >> lsfile
echo hi  >> lsfile
```

One of the main Linux philosophies is that everything is treated as a file including hardware devices. To interact with hardware devices, Linux interacts with device files which represent those hardware devices. This means that if we are able to redirect input or output from/to files then we actually do the same with devices. We can try this with device files that represent our terminals (screens).

Open two terminals (if using telnet then do two telnet connections).
Run the folloing command in both terminals:

```
tty
```

Record the ***pts*** numbers (you should have two, one form each terminal).
Assume the terminal you are working on has **/dev/pts/0** and the other terminal has /**dev**/**pts**/**1**.Type the following command:

```
echo hello
```

This will display the word hello on your current terminal (i.e. /**dev**/**pts**/**0**) which is the default. Now type the following command:

```
echo  hello  >  /dev/pts/1
```

What happened? Explain.

____

____

Error Redirection

Command output is sometimes mixed up with command errors since they are both sent to the screen by default. Run the following command:

```
cp
```

What did you get displayed?

____.

Is that output or error? ____________________________.

Now run the command:

```
cp > cpfile
```

What happened? ____.

Since the same message got displayed on the screen and was not sent to file ***cpfile*** then it must not be output. It is error.

To understand how to redirect errors, we should learn about file descriptors. There are three file descriptors used by programs to specify input, output, and error.

Standard input has file descriptor 0
Standard output has file descriptor 1
Standard error has file descriptor 2

There is no need to use the file descriptors 0 and 1 when redirecting input and output respectively since they use two different characters namely < and >.

To redirect error we need to use the (>) character so to distinguish it from redirecting error, we must specify the file descriptor before the > character as follows:

```
cp 2> cpfile
```

What happened now? ____.

Check the contents of file cpfile. What did you find?

____.

Redirecting output and error to different places may be very useful especially when dealing with commands that produce both at the same time. Try the following command:

```
find / -name passwd -print
```

What did you get? Was that output or error? ______________________________.

Now run the command as follows:

```
find / -name passwd -print 2> errors
```

What did you get now? ____.

Check file errors content.

Now run the command as follows:

```
find / -name passwd -print > output 2> error
```

What happened? ____.

Check both files output and error.

To append errors use (**2>>**).

Pipes

One of the main Linux philosophies is to have commands where each does one thing very well. For example, the ls command has so many options to display file information in so many different ways. Another philosophy that complements that is the ability to join different commands together in a chain to produce more powerful commands. This is usually done using pipes.

Run the following command:

```
cat /etc/passwd | grep yourusername | cut -d: -f5 | cut -d, -f1
```

What did you get? ____.

This command is made up of four different commands joined together using pipes (|). Pipes usually work with commands we call **filters**. They take input and filter it to produce a certain output. They usually do not change the original input source.
This is how the above command works:

"***cat /etc/passwd***" produces the passwd file (many lines) as output.
The **passwd** file is passed as input to the "***grep yourusername***" command which in turn filters that into a single line that contains your username. This line is then passed to the command "***cut -d: -f5***" which filters it to one -field (field five) (-f5) based on dividing fields by delimiter : (-d:). This output is then passed as input to the next cut command "***cut -d, -f1***" which filters it to get the first field (your full name) by cutting based on delimiter underscore (-d,). The output (your full name) is then displayed on the screen.

What command would you use to get your group number from /etc/passwd:

____.

What command would you use to get your login time from the who command?
(Hint: use the tr command with the squeeze option)

____.

What command would you use to get the default group name for any given user?

____.

Try the following command:

```
find / -name passwd -print | more
```

What happened? Why is the result of the command not filtered by more?

____

____.

How can we fix this?

____

____.