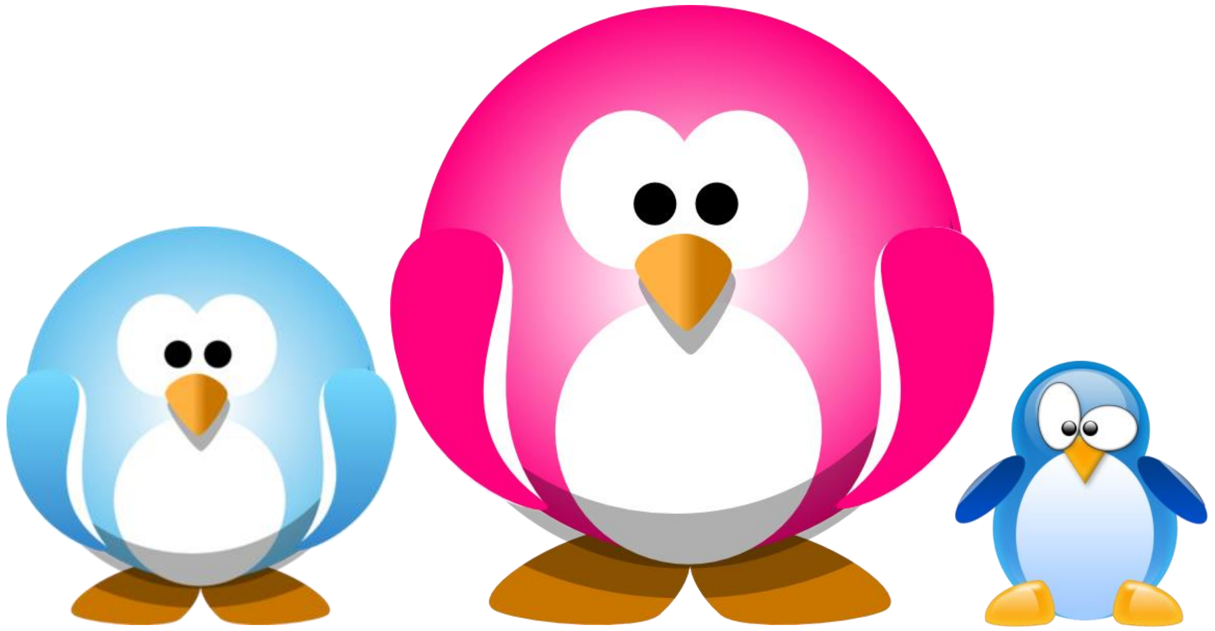
Comp311- Lab Linux

Lab 6

Shell Usage and Configuration (II)



Instructor :Murad Njoum

8/10/2021

I/O Redirection :

- Commands usually receive input and then produce output and error. By default the input is usually received **from the keyboard** and the output and errors are usually **both directed to the screen**.

- **Linux shells allow us to change those defaults and redirect input, output, and errors.**

Input Redirection

To understand **input redirection**, let us first use the **mail** command. The mail command is the default command used to send and receive mail on most UNIX based systems. To send email to another user simply use the command:

mail username (username@system if on another system)
You can try sending yourself an email by typing:
mail yourusername
subject:hello
This is my mail message
Goodbye
.

Cont..:

As you can see the mail asks you for a subject (title of message) and you end the mail by typing a dot **(.)** by itself on a line and then pressing enter.
To read your email, you can simply type:
mail
You will get the & sign. Type ? for help on how to use
(read/delete/save/reply/forward/…) the mail program. To quit just type **q** and Enter.
The input for the mail command was received from the keyboard (default). You can redirect the input such that it is received from a file. To do that use the (<) character as follows:
Create a **file called message** and type the following two lines inside:
This is my message file
Goodbye
Then save and quit

Now run the following command:
mail -s "hello" yourusername < message

The input in this case was redirected to come from **file message instead of the keyboard**

Translate

Another example is the **tr** (translate) command. This 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:

tr “a-z” “A-Z” < message
What was the output?

____

____

Recalling Commands from History

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).

Continue…

Open two terminals (if using telnet then do two telnet connections).

Run the command:
who

Record the **pts (pseudo terminal slave)** numbers (you should have two, one form each terminal). Assume the terminal you are working on has **pts/4** and the other terminal has ***pts/5*** (you need to use your numbers when testing).
tty:teltype terminal

Type the following command:
echo hello

This will display the word hello on your current terminal (i.e. pts/4) which is the default.
Now type the following command:
echo hello > /dev/pts/5
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 (stdin) /dev/stdin

Standard output has file descriptor 1 (stdout) /dev/stdout

Standard error has file descriptor 2 (stderr) /dev/stderr

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 >.**

Error redirection Continue..

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

find: search for files in a directory hierarchy

-name: Base of file name

-print :the full file name on the standard output

What did you get? Was that output or error?___________________________________

Error redirection Continue..

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? ____.

```
mnjoum:x:66:1000:Murad Njoum:/home/mnjoum:/bin/bash
```

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:

Pipes:

“**cat /etc/passwd**” produces the passwd file (many lines) as output.

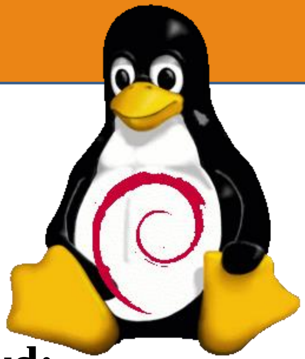
cat: concatenate files and print on the standard output cat message error

example : cat file1 file2 >outfile

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 first name) by cutting based on
delimiter underscore (-d_). The output (your first name) is then displayed on the screen.

Practice:



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

```
grep u1172111 /etc/passwd  | cut –d : -f4
```

____.

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

replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character, quoted from man of tr
____.

```
who|tr -s ' '|grep u1172111 |cut -d ' ' -f4
```

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

```
x=$(grep u1172111 /etc/passwd |cut -d : -f4)
grep $x /etc/group |cut -d : -f1
```

____

More Practice

Try the following command:

find / -name passwd -print | more

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

More doesn't work with error output. ___________________________

Filter (pipe) not pass error result

____.

How can we fix this?

____

find –name / passwd –print 2>&1|more

____.

Summary

- By default, most Linux programs expect input to come from disk file and output to go to the window
  - You can redirect input and output as follows
  - **command > file** – redirect output to file, if file already exists, override it
  - **command >> file** – redirect output to append to file, if file does not already exist, create it
  - **command < file** – redirect input to come from file (this is the typical case for most instructions, so is not particularly useful except when running shell scripts)
  - **command << string** – redirect input to come from keyboard, end the input **when "string"** is encountered
  - **command | command** – this is known as a pipe, take the output of one command and use it as input to the next

Thank You for attention !

Published By: Murad Njoum