COMP311
Linux OS Laboratory
Lab6:Shell Usage and Configuration (II)

By

Alaa’ Omar

Objectives

1

Understand and use shell input, output, and error redirection.

2

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.

I/O Redirections:

- **Input Redirection**
- **Output Redirection.**
- **Error Redirection.**

Input Redirection

When you login, the shell automatically sets standard input to the keyboard, standard output and standard error to the screen.

(read from keyboard → write to screen)

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
^D
cc:

Mail command

- 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:

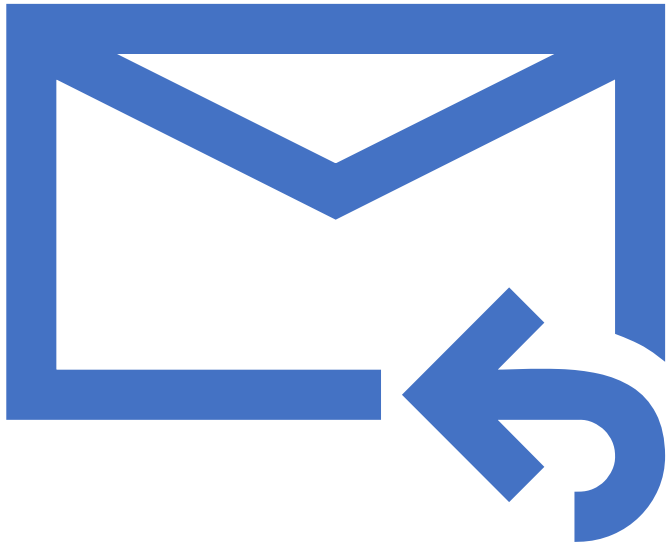
  mail
- You will get the & sign. Type **?** for help on how to use(read/delete/save/reply/forward/…) the mail program. To quit just type **q** and Enter.
- The input for the mail command was received from the keyboard **(default).** You can redirect the input such that it is received from a file. To do 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

Mail command

- 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 command

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?
```

Translate command

- 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

Output Redirection

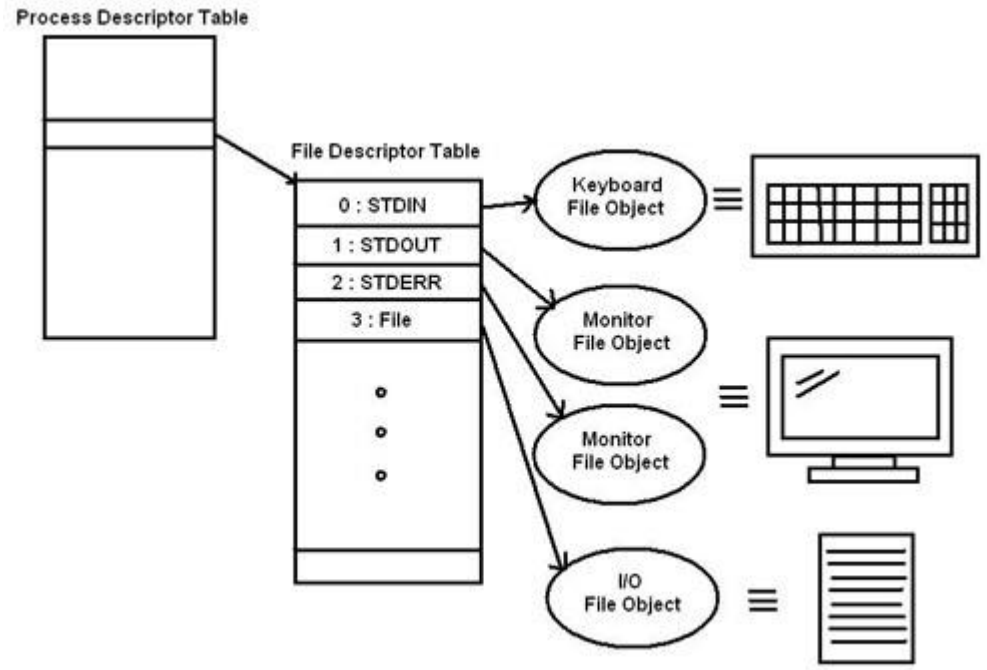
One of the main Linux philosophies is that everything is treated as a file including hardware devices. To interact with hardware devices, Linux interacts with device files which represent those hardware devices. This means that if we are able to redirect input or output from/to files then we do the same 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 command:
who
Record the **pts** 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).
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.

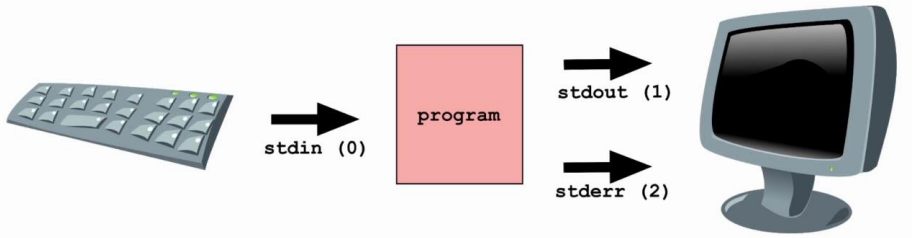
File Descriptors

A number to describe an open file or I/O resource in a system.

- Unique non-negative integer



http://web.cs.ucla.edu/classes/fall08/cs111/scribe/4/index.html



https://linuxmeerkat.wordpress.com/2011/12/02/file-descriptors-explained/

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

Error Redirection

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?

_____.

Error Redirection

find command is used to find a file by permission, type name, size, date and others.
Syntax: $ **find [path][expression][what to find]**

- **[path]: where to search the file, for example /etc**
- **[expression]: it might be option, action, TESTS, etc.**

Example:

$find . -name file1
$find . -readable
$find /home/alaa -writable
$find . -type f

Error Redirection

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

The command find / -name passwd -print is used to search for files with the name "passwd" starting from the root directory ("/") and print their paths.

Here's an explanation of each component of the command:

- find: This is the command used to search for files and directories.
- /: It specifies the starting point of the search, which is the root directory. The search will traverse the entire directory structure starting from the root.
- -name passwd: This is an option provided to the find command. It specifies the name of the file to search for. In this case, the name is "passwd".
- -print: This is another option provided to the find command. It instructs find to print the path of each file that matches the specified criteria.

When you run the command, it will search the entire file system, starting from the root directory, for files with the name "passwd". If any files with that name are found, their paths will be printed to the terminal.

Error Redirection

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>>**).

Combining standard output and standard error.

- To redirect the output to an output file and redirect the error to the same output file. The syntax to do so:

- find / -name passwd -print 2>&1

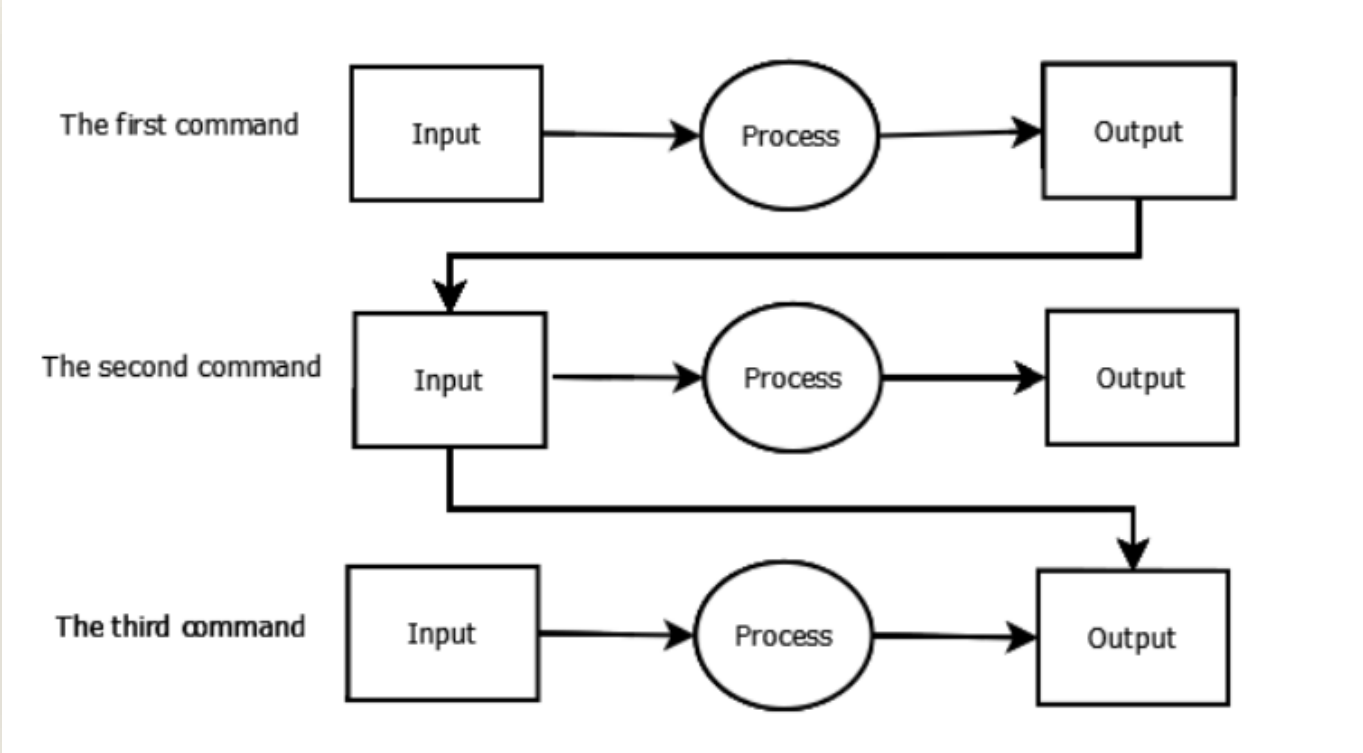
Redirection: Summaries

- **Bourne Shell family: redirection of standard I/O**

SYMBOL	ACTION
<	Redirect stdin (same as 0<)
>	Redirect stdout (same as 1>)
>>	Append stdout (same as 1>>)
2>	Redirect stderr
2>>	Append stderr
2&>1	Redirect stderr to stdout
|	Pipe stdout to another command
2&>1 |	Pipe stdout + stderr to another command

Pipes

Pipes connect the output from one command to the input of another command. In other words, instead of sending the output of a command to a destination file or device, pipes send that output to another command as input. This lets you have one command work on some data and then have the next command deal with the results.

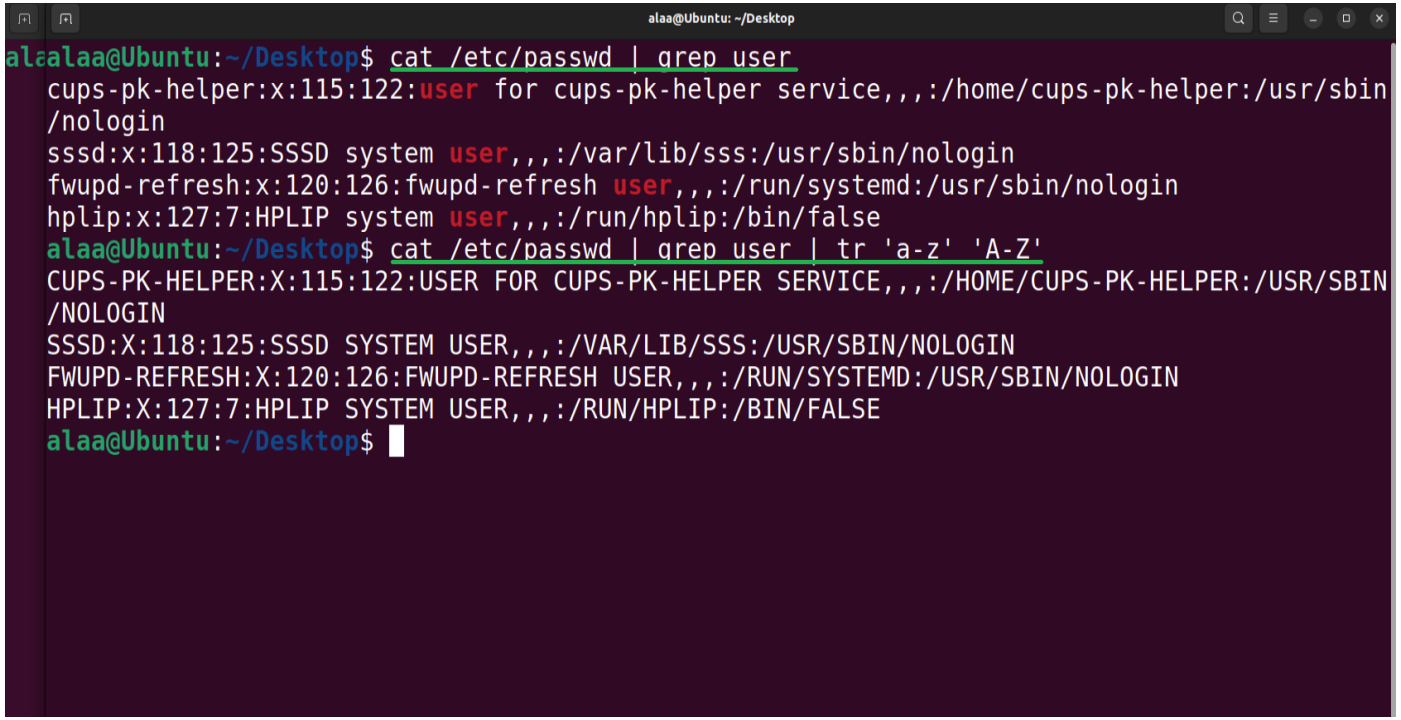


Example of Pipes

- $ ls ~ | sort

Example of Pipes

- $ cat /etc/passwd | grep user | tr ‘a-z’ ‘A-Z’



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

1. cat /etc/passwd: The cat command is used to display the contents of a file. Here, it is used to display the contents of the /etc/passwd file. This file contains information about user accounts on a Unix-like system.
2. |: The pipe symbol (|) is a command-line operator that allows the output of one command to be passed as input to another command.
3. grep yourusername: The grep command is used for searching patterns within files. In this case, it is used to search for the pattern "yourusername" in the output of the cat /etc/passwd command. Replace "yourusername" with the actual username you want to search for.
4. cut -d: -f5: The cut command is used to extract specific sections (columns) from input lines. Here, it is used to extract the fifth field/column from each line of input. The delimiter (-d) specified is :, and -f5 indicates that the fifth field should be extracted.
5. cut -d_ -f1: Another cut command is used here to further process the output from the previous cut command. This time, the delimiter (-d) specified is _, and -f1 indicates that the first field should be extracted. This is used to extract the part before the underscore character in the fifth field.

In summary, the command is extracting and displaying a specific portion of the fifth field in the /etc/passwd file, specifically the part before the underscore character, for the line that matches the given username.

What did you get? _____.

Pipes

- 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)

Pipes

who | tr -s ' ' | cut -d' ' -f4

Let's break down the command:

- who: The who command displays information about currently logged-in users.
- tr -s ' ': The tr command is used for character translation or deletion. The -s option squeezes multiple occurrences of the space character into a single space. This is useful to remove any extra spaces between columns in the output of the who command.
- cut -d' ' -f4: The cut command is used to extract specific sections (columns) from input lines. Here, it is used to extract the fourth field/column from each line of input. The delimiter (-d) specified is a space character (' '), and -f4 indicates that the fourth field should be extracted. In the context of the who command, the fourth field represents the login time.

By combining these commands, you can extract and display the login time from the who command's output.

Pipes

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?

More doesn't work with error output

find / -name passwd -print 2>error | more (only shows output but not errors)

Find / -name passwd –print 2>&1 | more

Pipes

The command **find / -name passwd -print 2>error | more** is used to search for files or directories named "**passwd**" starting from the current directory and display the results page by page using the more command.

Here's a breakdown of the command:

- find: The command for searching files and directories recursively.
- / -name passwd: The option -name specifies that we want to search for files or directories with the exact name "passwd". The / before "passwd" indicates that we are searching from the root directory.
- -print: The action to be performed when a matching file or directory is found. In this case, it prints the path of the file or directory.
- 2>error: Redirects the error output (stderr) to the error file while printing the output paths on screen.
- |: The pipe symbol redirects the standard output of the preceding command (find -name /passwd -print 2>&1) to the next command (more).
- more: A command used for paginating text output. It displays the output page by page, allowing the user to scroll through it.

So, when you run the command find -name /passwd -print 2>&1 | more, it searches for files or directories named "passwd" starting from the root directory, prints the results, redirects any error messages to the standard output, and displays the output page by page using the more command

Pipes “Practice”

- Write a command to **extract the login date** from the **who** command in the following format (YYYY/mm/dd) e.g., 2023/05/22 instead of 2023-05-22

The End