Chapter 7

Shell Input/output Redirections

School of Computer Science,
MIT-World Peace University,
Pune, Maharashtra, India.

Amol D. Vibhute (PhD)
Assistant Professor

Email ID:- amol.vibhute@mitwpu.edu.in

Roadmap of Chapter

- Output Redirection
- Input Redirection
- Here Document
- Discard the output
- Redirection Commands.

Introduction:

- Redirection is a feature in Linux/Unix such that when executing a command, we can change the standard input/output devices.
- Most Unix system commands take input from our terminal and send the resulting output back to our terminal.
- The basic workflow of any Unix command is that it takes an input and give an output.
- A command normally reads its input from the standard input, which happens to be our terminal by default.
- Similarly, a command normally writes its output to standard output, which is again our terminal by default.
  - The standard input (stdin) device is the keyboard.
  - The standard output (stdout) device is the screen.
- With redirection, the above standard input/output can be changed.

Output Redirection:

- The output from a command normally intended for standard output can be easily diverted to a file instead. This capability is known as output redirection.
- If the notation **>** file is added to any command that normally writes its output to standard output, the output of that command will be written to file instead of your terminal.
- Check the following who command which redirects the complete output of the command in the user's file.
  - $ who > users
- Notice that no output appears at the terminal. This is because the output has been redirected from the default standard output device (the terminal) into the specified file. We can check the users file for the complete content −
  - $ cat users
- If a command has its output redirected to a file and the file already contains some data, that data will be lost.

Cont.…

- Consider the following example −
  - $ echo “Hello Unix World” > users
  - $ cat users
  - The output will be- Hello Unix World.
- We can use **>> operator** to append the output in an existing file as follows −
  - $ echo “Unix of Ty. Bsc” >> users
  - $ cat users
  - The output will be-
  - Hello Unix World
  - Unix of Ty. Bsc
- If we only use the **> operator** with same file name, then the output will be changed while removing the previous contents.

Input Redirection:

- Just as the output of a command can be redirected to a file, so can the input of a command be redirected from a file.
- As the **greater-than character >** is used for output redirection, the **less-than character <** is used to redirect the input of a command.
- The commands that normally take their input from the standard input can have their input redirected from a file in this manner.
- For example, to count the number of lines in the file users generated above, we can execute the command as follows-
  - $ wc -l users
  - 2 users
- Upon execution, we will receive the following output. We can count the number of lines in the file by redirecting the standard input of the **wc command** from the file users-
  - $ wc -l < users
  - 2
- Note that there is a difference in the output produced by the two forms of the wc command. In the first case, the name of the file users is listed with the line count; in the second case, it is not.
- In the first case, wc knows that it is reading its input from the file users. In the second case, it only knows that it is reading its input from standard input, so it does not display file name.

Here Document:

- A **here document** is used to redirect input into an interactive shell script or program.
- We can run an interactive program within a shell script without user action by supplying the required input for the interactive program, or interactive shell script.
- The general form for a **here document** is −
  - command << delimiter
  - document
  - delimiter
- Here the shell interprets the **<< operator** as an instruction to read input until it finds a line containing the specified delimiter.
- All the input lines up to the line containing the **delimiter** are then fed into the standard input of the command.
- The delimiter tells the shell that the **here document** has completed.
- Without it, the shell continues to read the input forever. The delimiter must be a single word that does not contain spaces or tabs.

Cont....

- Following is the input to the command **wc -l** to count the total number of lines −
  - $wc -l << EOF
  - This is a simple lookup program
  - for good (and bad) restaurants
  - in Pune.
  - And Mumbai
  - EOF
  - 4
  - $
- We can use the **here** document to print multiple lines using the script as follows −
  - #!/bin/sh
  - cat << EOF
  - This is a simple lookup program
  - for good (and bad) restaurants
  - in Pune
  - And Mumbai.
  - EOF
- Upon execution, we will receive the following result −
  - This is a simple lookup program
  - for good (and bad) restaurants
  - in Pune
  - And Mumbai.

Discard the output:

- Sometimes we will need to execute a command, but we don't want the output displayed on the screen.
- In such cases, we can discard the output by redirecting it to the file /dev/null −
  - $ command > /dev/null
- Here **command** is the name of the command we want to execute. The file /dev/null is a special file that automatically discards all its input.
- To discard both output of a command and its error output, use standard redirection to redirect STDERR to STDOUT −
  - $ wc –l abc.txt > /dev/null 2>&1
- Here **2** represents **STDERR** and **1** represents **STDOUT**.

Redirection Commands:

- Following is a complete list of commands which you can use for redirection –
- Note that the file descriptor **0** is normally standard input (STDIN), **1** is standard output (STDOUT), and **2** is standard error output (STDERR).

Sr. No.	Command & Description
1.	**pgm > file:** Output of pgm is redirected to file
2.	**pgm < file:** Program pgm reads its input from file
3.	**pgm >> file:** Output of pgm is appended to file
4.	**n >& m:** Merges output from stream **n** with stream **m**
5.	**n <& m:** Merges input from stream **n** with stream **m**

References:

- Unix Shell Programming: Yashwant Kanitkar, BPB Publications, New Delhi.

Thank You !!!