**Linux Fundamentals**

**What is Shell ?**

It is the Command interpreter application that accept the commands you type and then run programs in response to those command/instructions

**What is bash?**

Is the default shell command interpreter in Linux. The default linux shell is Bash which stands for “Bourne-Again Shell”

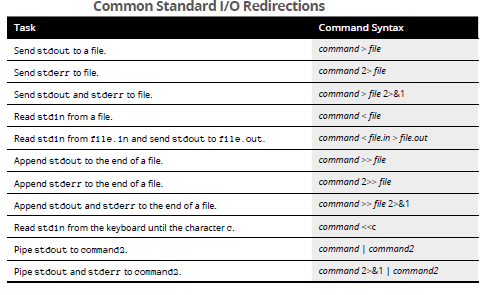
**What is PIPE?**

That vertical bar (|) is known as a pipe because it acts as a conduit (think of a water pipe) between the two programs. The output of the first command is fed into the input of the second one.

**Controlling command input and output**

Most Linux commands have a common feature: They always read from the standard input (usually, the keyboard) and write to the standard output (usually, the screen). Error messages are sent to the standard error (usually, to the screen as well). These three devices are often referred to as stdin, stdout, and stderr.

You can make a command get its input from a file and then send its output to another file. Just so you know, the highfalutin term for this feature is input and output (I/O) redirection. Table below shows the syntax of common I/O redirection commands, and the next few sections explain how to use some of these commands.



**Getting command input from a file**

If you want a command to get its instructions by reading from a file, you can redirect the standard input to come from that file instead of from the keyboard. The command

sort < /etc/passwd

displays a sorted list of the lines in the /etc/passwd file. In this case, the less than sign (<) redirects stdin so that the sort command reads its input from the /etc/passwd file.

**Saving command output in a file**

To save the output of a command in a file, redirect the standard output to a file. Type **cd** to change to your home directory and then type the following command:

grep typedef /usr/include/\* > typedef.out

This command searches all files in the /usr/include directory for the occurrence of the text typedef and then saves the output in a file called typedef.out. The greater-than sign (>) redirects stdout to a file. This command also illustrates another feature of bash: When you use an asterisk (\*), bash replaces the asterisk with a list of all filenames in the specified directory. Therefore, /usr/include/\* means all the files in the /usr/include directory.

If you want to append a command’s output to the end of an existing file instead of saving the output in a new file, use two greater-than signs (>>), like this:

command >> filename

Another interesting way to send stdout to a file is to use the cat command to prepare small text files quickly. Suppose that you want to create a new text file to store lines of text you type until you type **ZZ** and press Enter. Here’s how you can accomplish that task:

cat <<ZZ > input.txt

After you type this command, you can keep typing lines and then type **ZZ** on a line when you finish. Everything you type is saved in the file input.txt.

**Saving error messages in a file**

Sometimes, when you type a command, it generates a lot of error messages that scroll by so fast that you can’t tell what’s going on. One way to see all the error messages is to save them in a file so that you can see what the heck happened. You can do that by redirecting stderr to a file.

Type the following command:

find / -name COPYING -print 2> finder

This command looks through the file system for files named COPYING and saves all the error messages (if any) in the finderr file. The number 2 followed by the greater-than sign (2>) redirects stderr to a file.

If you want to discard the error messages instead of saving them in a file, use / dev/null as the filename, like this:

find / -name COPYING -print 2> /dev/null

That /dev/null is a special file (often called the *bit bucket* and sometimes glorified as the *Great Bit Bucket in the Sky)* that simply discards whatever it receives. Now you know what it means when you hear a phrase such as “Your mail probably ended up in the bit bucket.”

**PIPE**

Connecting commands together with pipes is one of the most powerful features of Linux. Linux does not have a command to count the number of files in a directory but it does have one command to list the files (ls) and a second command (wc) to count the number of lines in a list. You could therefore use a pipe (|) to glue ls and wc together:

ls | wc-l

Note that we are using the-l (that is a hyphen followed by the lower-case letter l) argument or option for wc.

**ANSI Escape Codes**

ANSI escape sequences are non-printed text that is interpreted to change the format of some text.

Their support depends on the terminal, but the color control sequences are commonly supported

echo -e "\e[01;32m"; $(date +%R); echo -e "\e[00m"

For more info:

<https://www.youtube.com/watch?v=UaRDNlprbfk>