Input Output Redirection in Linux

•Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices.

•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.

For redirection, meta characters are used.

Redirection can be into a **file** (shell meta characters are angle **brackets** '<', '>') or a **program** (shell meta characters are **pipe**symbol '|').

Redirection Into A File

Each stream uses redirection commands.

Single bracket '>' or double bracket '>>' can be used to redirect standard output.

If the target file doesn't exist, a new file with the same name will be created.

Overwrite
Commands with a single bracket **'>' overwrite** existing file content.

➢: standard output

< : standard input

Note: Writing **'1>'** or **'>'** and **'0<'** or **'<'** is same thing.

Syntax:
cat **> <fileName>**

Example:
cat **>** sample.txt

```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ cat > sample.txt
a
b
c
sssit@JavaTpoint:~$ cat  sample.txt
a
b
c
sssit@JavaTpoint:~$ cat > sample.txt
d
e
f
sssit@JavaTpoint:~$ cat  sample.txt
d
e
f
sssit@JavaTpoint:~$
```

Look at the above snapshot,
command "cat > sample.txt" has created 'sample.txt' with content 'a, b, c'.

Same file 'sample.txt' is created again with command **"cat > sample.txt"**

and this time it overwrites earlier file content and only displays 'd, e, f '.

Append Commands with a double bracket **'>>' do not overwrite** the existing file content.

>> - standard output

<< - standard input

Syntax:

cat **>> <fileName>**

Example:

cat **>>** sample.txt

```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ cat >> sample.txt
a
b
c
sssit@JavaTpoint:~$ cat sample.txt

a
b
c
sssit@JavaTpoint:~$ cat >> sample.txt
d
e
f
sssit@JavaTpoint:~$ cat sample.txt

a
b
c
d
e
f
sssit@JavaTpoint:~$
```

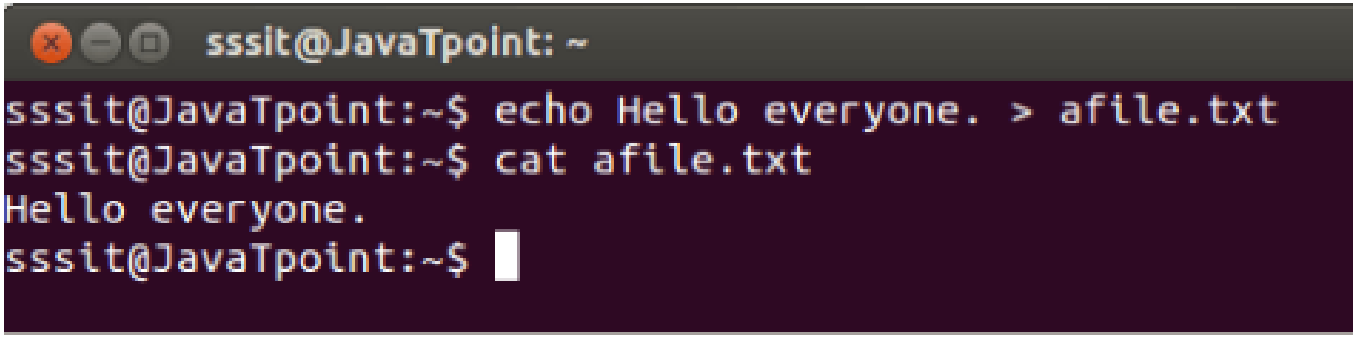
Look at the above snapshot,
here again we have created two files with the same name using '>>' in command **"cat >> sample.txt"**.

But this time, content doesn't overwrite and everything is displayed.

Linux Output Redirection

The stdout is redirected with a '>' greater than sign. When shell meets the '>' sign, it will clear the file (as you already know).

Example:
echo Hello everyone. **>** afile.txt



Look at the above snapshot, greater than sign '>' redirects the command 'echo' output into a file 'afile.txt'.

Piping in Unix

A pipe is a form of redirection that is used in Linux to send the output of one command/program/process to another

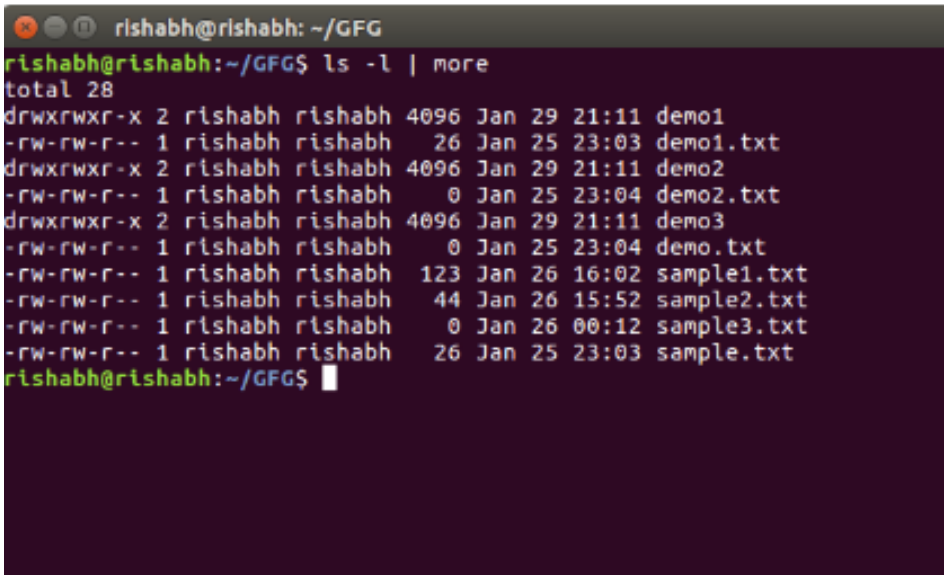
Pipe is used to combine two or more commands, by using the pipe character **'|'**.

Pipes are unidirectional **i.e data flows from left to right through the pipeline.**

Syntax :
command_1 | command_2 | command_3 | .... | command_N

Example :
1. Listing all files and directories and give it as input to more command.

$ ls -l | more



2. Use sort and uniq command to sort a file and print unique values.

$ sort record.txt | uniq

This will sort the given file and print the unique values only.

Output :

```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ cat result.txt
Rajat Dua               ECE     9.1
Rishabh Gupta           CSE     8.4
Prakhar Agrawal         CSE     9.7
Aman Singh              ME      7.9
Rajat Dua               ECE     9.1
Rishabh Gupta           CSE     8.4
Aman Singh              ME      7.9
Naman Garg              CSE     9.4
rishabh@rishabh:~/GFG$ sort result.txt | uniq
Aman Singh              ME      7.9
Naman Garg              CSE     9.4
Prakhar Agrawal         CSE     9.7
Rajat Dua               ECE     9.1
Rishabh Gupta           CSE     8.4
rishabh@rishabh:~/GFG$
```

Metacharacters: These are the special characters that are first interpreted by the shell before passing the same to the command. They are also known as shell wildcards.

$ Variable Substitution or expand the value of Variable.

> used for Output Redirection.

>> used for Output Redirection to append.

< Input redirection.

<< used for input redirection and is also known as here document.

***** Match any number of characters, Substitution wildcard for zero or more characters

? Match one character, Substitution wildcard for 1 character

[] Match range of characters, Substitution wildcard for any character between brackets

`cmd` Replace cmd with the command to execute and will execute that, Substitution wildcard for command execution

$(cmd) Replace cmd with the command to execute and will execute that, Substitution wildcard for command execution

| Pipe is a Redirection to send the output of one command/program/process to another command/program/process for further processing.

; Command separator is used to execute 2 or more commands with one statement.

|| OR conditional execution of the commands.

&& AND conditional execution of the commands.

() Groups the command in to one output stream.

& executes command in the background and will display the assigned Pid.

to comment something.

$ To expand the value of a variable.

**** used to escape the interpretation of a character or to prevent that.

A **Backslash('\')** is the bash escape character.

Single Quotes(") are used to preserve the literal value of each character within the quotes.

Double Quotes("") preserves the literal value of all characters within the quotes, with the exception of '$', '`', '\', and when history expansion is enabled, '!'.

Back Quotes(") are used to execute a command.