

Piping in Unix or Linux

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing. The Unix/Linux systems allow stdout of a command to be connected to stdin of another command. You can make it do so by using the pipe character '|'.

Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes. The command line programs that do the further processing are referred to as filters.

This direct connection between commands/ programs/ processes allows them to operate simultaneously and permits data to be transferred between them continuously rather than having to pass it through temporary text files or through the display screen.

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

Syntax :

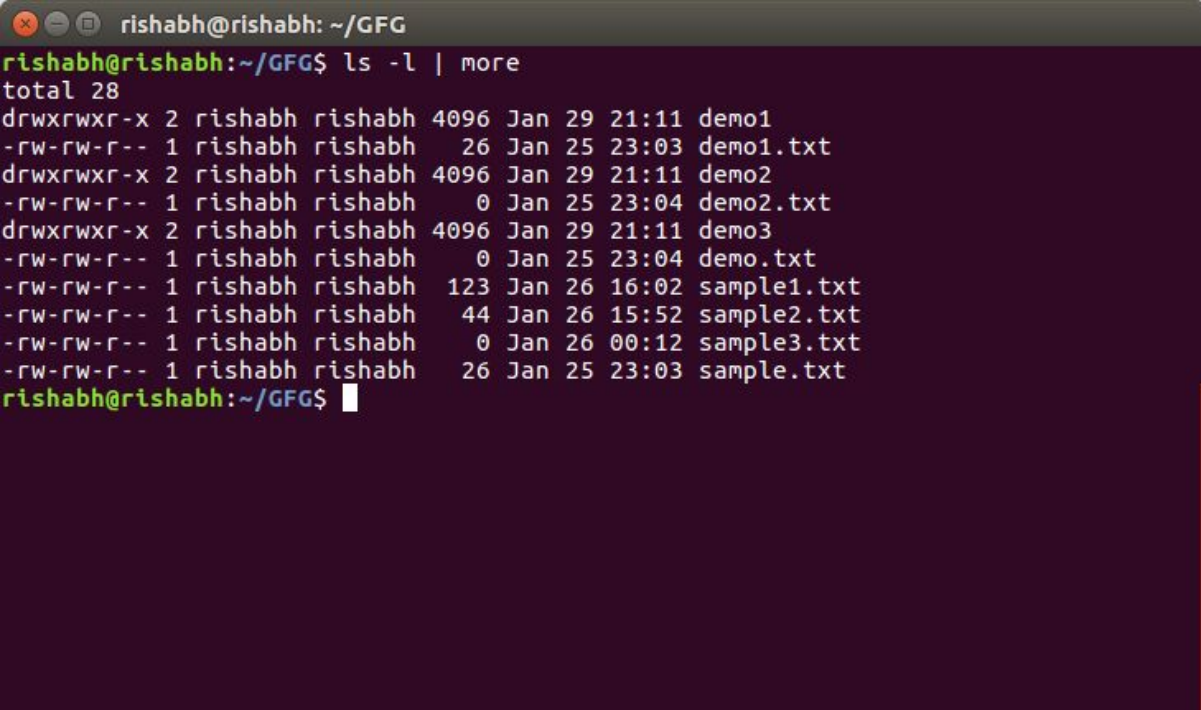
```
command_1 | command_2 | command_3 | ..... | command_N
```

Example :

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

```
$ ls -l | more
```

Output :

A terminal window with a dark purple background. The title bar shows 'rishabh@rishabh: ~/GFG'. The prompt is 'rishabh@rishabh:~/GFG\$'. The command 'ls -l | more' has been entered. The output shows the directory listing for the current directory, with the first line 'total 28' on a new line. The following lines list files and directories with their permissions, sizes, dates, and names. The output is truncated by the 'more' command, with a cursor visible at the end of the last line.

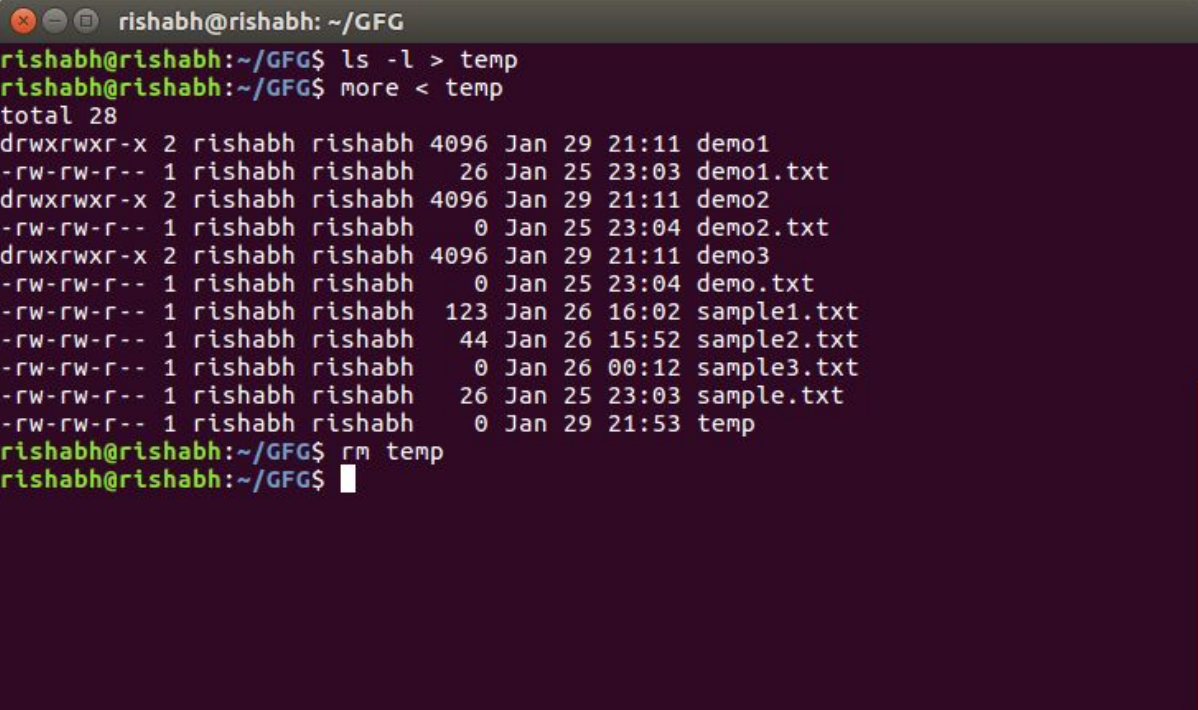
```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ ls -l | more
total 28
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo1
-rw-rw-r-- 1 rishabh rishabh  26 Jan 25 23:03 demo1.txt
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo2
-rw-rw-r-- 1 rishabh rishabh   0 Jan 25 23:04 demo2.txt
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo3
-rw-rw-r-- 1 rishabh rishabh   0 Jan 25 23:04 demo.txt
-rw-rw-r-- 1 rishabh rishabh  123 Jan 26 16:02 sample1.txt
-rw-rw-r-- 1 rishabh rishabh   44 Jan 26 15:52 sample2.txt
-rw-rw-r-- 1 rishabh rishabh   0 Jan 26 00:12 sample3.txt
-rw-rw-r-- 1 rishabh rishabh  26 Jan 25 23:03 sample.txt
rishabh@rishabh:~/GFG$
```

The more command takes the output of `$ ls -l` as its input. The net effect of this command is that the output of `ls -l` is displayed one screen at a time. The pipe acts as a container which takes the output of `ls -l` and gives it to more as input. This command does not use a disk to connect standard output of `ls -l` to the standard input of more because pipe is implemented in the main memory.

In terms of I/O redirection operators, the above command is equivalent to the following command sequence.

```
$ ls -l -> temp
more -> temp (or more temp)
[contents of temp]
rm temp
```

Output :



```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ ls -l > temp
rishabh@rishabh:~/GFG$ more < temp
total 28
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo1
-rw-rw-r-- 1 rishabh rishabh  26 Jan 25 23:03 demo1.txt
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo2
-rw-rw-r-- 1 rishabh rishabh   0 Jan 25 23:04 demo2.txt
drwxrwxr-x 2 rishabh rishabh 4096 Jan 29 21:11 demo3
-rw-rw-r-- 1 rishabh rishabh   0 Jan 25 23:04 demo.txt
-rw-rw-r-- 1 rishabh rishabh  123 Jan 26 16:02 sample1.txt
-rw-rw-r-- 1 rishabh rishabh   44 Jan 26 15:52 sample2.txt
-rw-rw-r-- 1 rishabh rishabh   0 Jan 26 00:12 sample3.txt
-rw-rw-r-- 1 rishabh rishabh  26 Jan 25 23:03 sample.txt
-rw-rw-r-- 1 rishabh rishabh   0 Jan 29 21:53 temp
rishabh@rishabh:~/GFG$ rm temp
rishabh@rishabh:~/GFG$
```

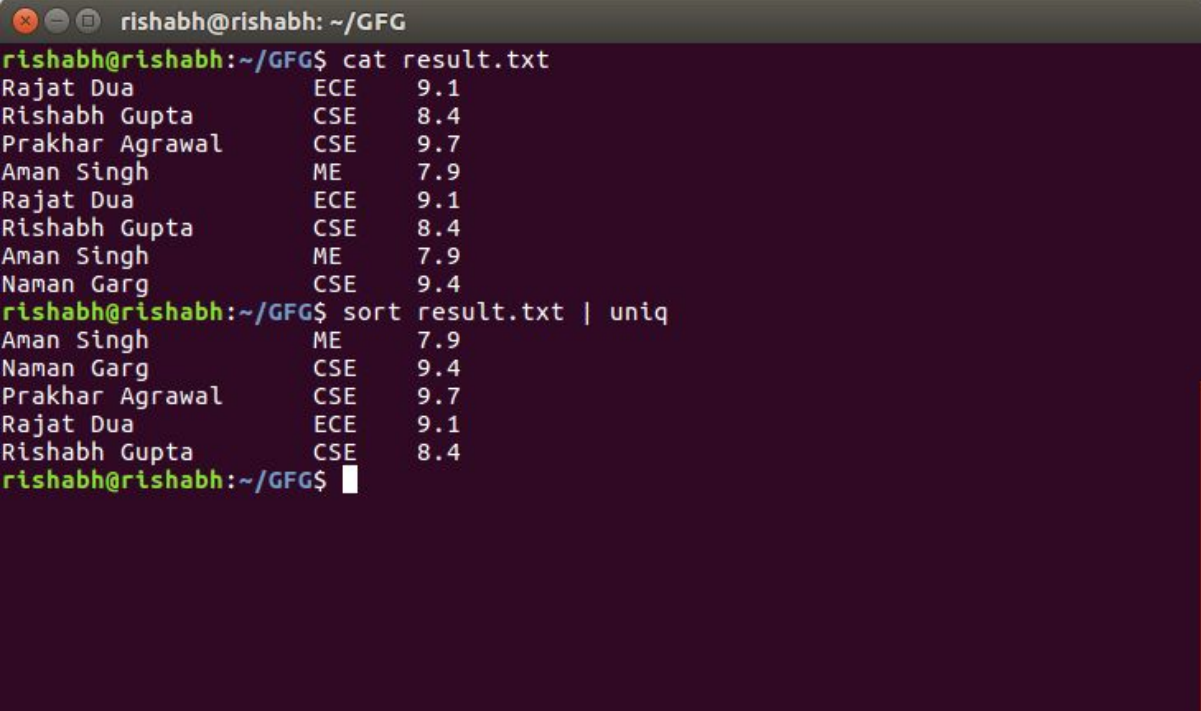
Output of the above two commands is same.

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 :



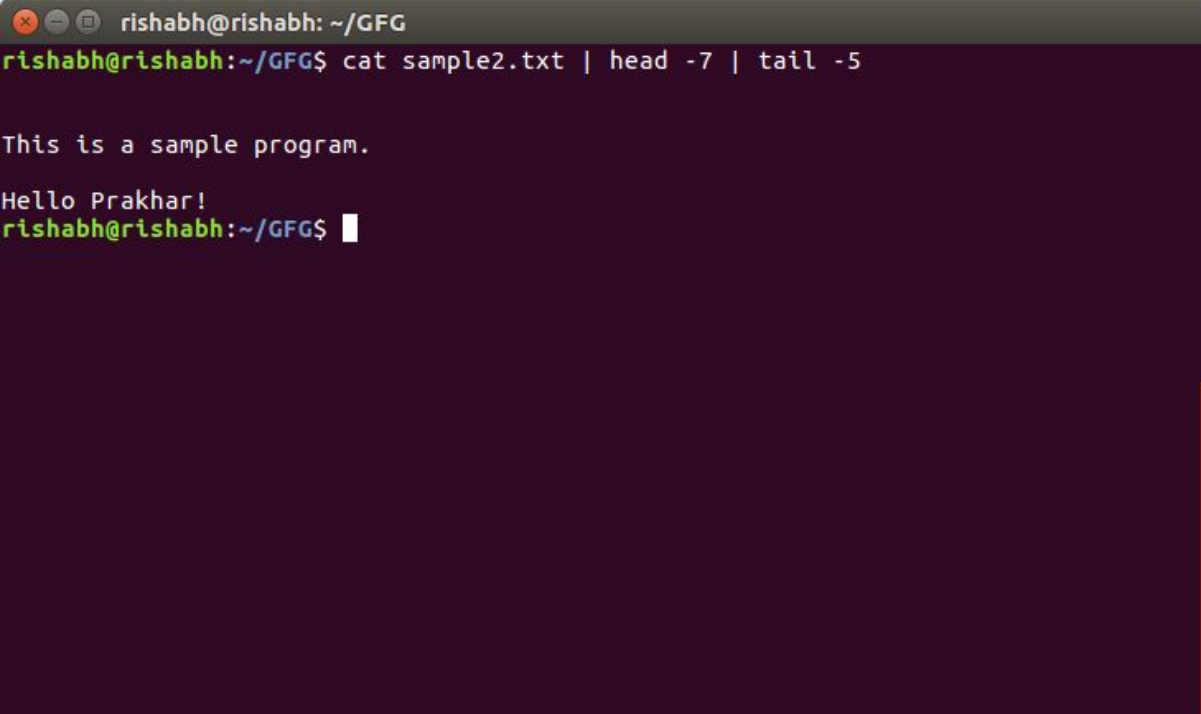
```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ cat result.txt
Rajat Dua          ECE      9.1
Rishabh Gupta      CSE       8.4
Prakhar Agrawal    CSE       9.7
Aman Singh         ME        7.9
Rajat Dua          ECE      9.1
Rishabh Gupta      CSE       8.4
Aman Singh         ME        7.9
Naman Garg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

3. Use head and tail to print lines in a particular range in a file.

```
$ cat sample2.txt | head -7 | tail -5
```

This command select first 7 lines through (head -7) command and that will be input to (tail -5) command which will finally print last 5 lines from that 7 lines.

Output :

A terminal window with a dark purple background. The title bar shows 'rishabh@rishabh: ~/GFG'. The prompt is 'rishabh@rishabh:~/GFG\$'. The command entered is 'cat sample2.txt | head -7 | tail -5'. The output displayed is 'This is a sample program.' followed by 'Hello Prakhar!' on the next line. The prompt 'rishabh@rishabh:~/GFG\$' is shown again with a cursor.

```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ cat sample2.txt | head -7 | tail -5

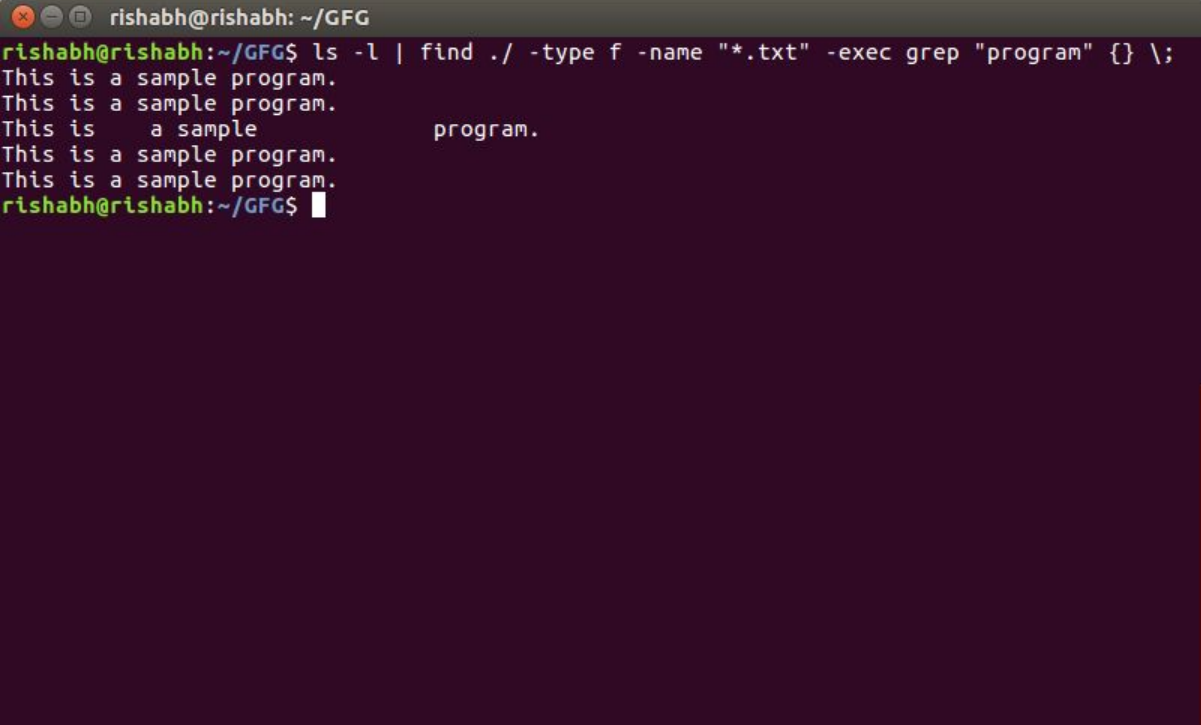
This is a sample program.
Hello Prakhar!
rishabh@rishabh:~/GFG$
```

4. Use ls and find to list and print all lines matching a particular pattern in matching files.

```
$ ls -l | find ./ -type f -name "*.txt" -exec grep "program"
{} \;
```

This command select files with **.txt** extension in the given directory and search for pattern like “program” in the above example and print those ine which have program in them.

Output :

A terminal window with a dark purple background. The title bar shows 'rishabh@rishabh: ~/GFG'. The prompt is 'rishabh@rishabh:~/GFG\$'. The command entered is 'ls -l | find ./ -type f -name "*.txt" -exec grep "program" {} \;'. The output consists of five lines: 'This is a sample program.', 'This is a sample program.', 'This is a sample program.', 'This is a sample program.', and 'This is a sample program.'. The prompt is now 'rishabh@rishabh:~/GFG\$' with a cursor.

```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ ls -l | find ./ -type f -name "*.txt" -exec grep "program" {} \;
This is a sample program.
This is a sample program.
This is a sample program.
This is a sample program.
This is a sample program.
rishabh@rishabh:~/GFG$
```

5. Use cat, grep, tee and wc command to read the particular entry from user and store in a file and print line count.

```
$ cat result.txt | grep "Rajat Dua" | tee file2.txt | wc -l
```

This command select **Rajat Dua** and store them in file2.txt and print total number of lines matching **Rajat Dua**

Output :

```
rishabh@rishabh: ~/GFG
rishabh@rishabh:~/GFG$ cat result.txt | grep "Rajat Dua" | tee file2.txt | wc -l
2
rishabh@rishabh:~/GFG$ cat file2.txt
Rajat Dua      ECE    9.1
Rajat Dua      ECE    9.1
rishabh@rishabh:~/GFG$
```