

# UNIX / LINUX - PIPES AND FILTERS

<http://www.tutorialspoint.com/unix/unix-pipes-filters.htm>

Copyright © tutorialspoint.com

## Advertisements

In this chapter, we will discuss in detail about pipes and filters in Unix. You can connect two commands together so that the output from one program becomes the input of the next program. Two or more commands connected in this way form a pipe.

To make a pipe, put a vertical bar (**&verbar;**) on the command line between two commands.

When a program takes its input from another program, it performs some operation on that input, and writes the result to the standard output. It is referred to as a *filter*.

## The grep Command

The grep command searches a file or files for lines that have a certain pattern. The syntax is –

```
$grep pattern file(s)
```

The name "**grep**" comes from the ed (a Unix line editor) command **g/re/p** which means “globally search for a regular expression and print all lines containing it”.

A regular expression is either some plain text (a word, for example) and/or special characters used for pattern matching.

The simplest use of grep is to look for a pattern consisting of a single word. It can be used in a pipe so that only those lines of the input files containing a given string are sent to the standard output. If you don't give grep a filename to read, it reads its standard input; that's the way all filter programs work –

```
$ls -l | grep "Aug"
-rw-rw-rw-  1 john  doc      11008 Aug  6 14:10 ch02
-rw-rw-rw-  1 john  doc       8515 Aug  6 15:30 ch07
-rw-rw-r--  1 john  doc       2488 Aug 15 10:51 intro
-rw-rw-r--  1 carol doc       1605 Aug 23 07:35 macros
$
```

There are various options which you can use along with the **grep** command –

Sr.No.	Option & Description
1	<b>-v</b>  Prints all lines that do not match pattern.
2	<b>-n</b>  Prints the matched line and its line number.
3	<b>-l</b>  Prints only the names of files with matching lines (letter "l")
4	<b>-c</b>

	Prints only the count of matching lines.
5	<b>-i</b>  Matches either upper or lowercase.

Let us now use a regular expression that tells grep to find lines with "**carol**", followed by zero or other characters abbreviated in a regular expression as ".\*"), then followed by "Aug".–

Here, we are using the **-i** option to have case insensitive search –

```
$ls -l | grep -i "carol.*aug"
-rw-rw-r--  1 carol doc      1605 Aug 23 07:35 macros
$
```

### The sort Command

The **sort** command arranges lines of text alphabetically or numerically. The following example sorts the lines in the food file –

```
$sort food
Afghani Cuisine
Bangkok Wok
Big Apple Deli
Isle of Java

Mandalay
Sushi and Sashimi
Sweet Tooth
Tio Pepe's Peppers
$
```

The **sort** command arranges lines of text alphabetically by default. There are many options that control the sorting –

Sr.No.	Description
1	<b>-n</b>  Sorts numerically (example: 10 will sort after 2), ignores blanks and tabs.
2	<b>-r</b>  Reverses the order of sort.
3	<b>-f</b>  Sorts upper and lowercase together.
4	<b>&amp;plus;x</b>

	Ignores first <b>x</b> fields when sorting.
--	---

More than two commands may be linked up into a pipe. Taking a previous pipe example using **grep**, we can further sort the files modified in August by the order of size.

The following pipe consists of the commands **ls**, **grep**, and **sort** –

```
$ls -l | grep "Aug" | sort +4n
-rw-rw-r-- 1 carol doc      1605 Aug 23 07:35 macros
-rw-rw-r-- 1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-rw- 1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-rw- 1 john  doc     11008 Aug  6 14:10 ch02
$
```

This pipe sorts all files in your directory modified in August by the order of size, and prints them on the terminal screen. The sort option `&plus;4n` skips four fields (fields are separated by blanks) then sorts the lines in numeric order.

## The **pg** and **more** Commands

A long output can normally be zipped by you on the screen, but if you run text through **more** or use the **pg** command as a filter; the display stops once the screen is full of text.

Let's assume that you have a long directory listing. To make it easier to read the sorted listing, pipe the output through **more** as follows –

```
$ls -l | grep "Aug" | sort +4n | more
-rw-rw-r-- 1 carol doc      1605 Aug 23 07:35 macros
-rw-rw-r-- 1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-rw- 1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-r-- 1 john  doc     14827 Aug  9 12:40 ch03
.
.
.
-rw-rw-rw- 1 john  doc     16867 Aug  6 15:56 ch05
--More-- (74%)
```

The screen will fill up once the screen is full of text consisting of lines sorted by the order of the file size. At the bottom of the screen is the **more** prompt, where you can type a command to move through the sorted text.

Once you're done with this screen, you can use any of the commands listed in the discussion of the **more** program.