Confidentiality Statement

Tata Code of Conduct

We, in our dealings, are self-regulated by a Code of Conduct as enshrined in the Tata Code of Conduct. We request your support in helping us adhere to the Code in letter and spirit. We request that any violation or potential violation of the Code by any person be promptly brought to the notice of the Local Ethics Counselor or the Principal Ethics Counselor or the CEO of TCS. All communication received in this regard will be treated and kept as confidential.

TATA CONSULTANCY SERVICES

vILP – Level 2

Content Manual

Version 1.1

December 2014

(ILP Guwahati)

Table of Content

1.8.grep

grep command allows you to search one file or multiple files for lines that contain a pattern. Exit status is 0 if matches were found, 1 if no matches were found, and 2 if errors occurred.

grep search the target file(s) for occurrences of pattern, where pattern may be literal text or a Regular Expression. It works as a filter on stdout, as in a pipe if no target file(s) specified,

Syntax:

grep pattern [file...]

Examples: Suppose we have created the following demo file.

```
$ cat demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.

Two lines above this line is empty.
And this is the last line.
```

1.8.1. Search for the given string in a single file

The basic usage of grep command is to search for a specific string in the specified file as shown below.

```
$ grep "this" demo_file
this line is the 1st lower case line in this file.
Two lines above this line is empty.
```

1.8.2. Checking given string in multiple files

A basic usage of grep command is searching for the given string in multiple files. For this example, let us copy the demo_file to demo_file1. The grep output will also include the file name in front of the line that matched the specific pattern as shown below. When the Linux shell sees the meta character, it does the expansion and gives all the files as input to grep.

```
$ cp demo_file demo_file1    //to copy the file demo_file to demo_file1
$ grep "this" demo_*
demo_file:this line is the 1st lower case line in this file.
demo_file:Two lines above this line is empty.
demo_file:And this is the last line.
demo_file1:this line is the 1st lower case line in this file.
demo_file1:Two lines above this line is empty.
```

```
demo_file1:And this is the last line.
```

1.8.3. *Case insensitive search*

We can use grep to search for the given string/pattern case insensitively. So it matches all the words such as “the”, “THE” and “The” case insensitively as shown below.

```
$ grep -i “the” demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.
And this is the last line.
```

1.8.4. *Match regular expression in files*

This is a very powerful feature of grep . In the following example, it searches for all the pattern that starts with “lines” and ends with “empty” with anything in-between. i.e To search “lines[anything in-between]empty” in the demo_file.

```
$ grep "lines.*empty" demo_file
Two lines above this line is empty.
```

A regular expression may be followed by one of several repetition operators:

- ? The preceding item is optional and matched at most once.
- The preceding item will be matched zero or more times.
- + The preceding item will be matched one or more times.
- {n} The preceding item is matched exactly n times.
- {n,} The preceding item is matched n or more times.
- {,m} The preceding item is matched at most m times.
- {n,m} The preceding item is matched at least n times, but not more than m times.

1.8.5. *Checking for full words*

To search for a word, and to avoid it to match the substrings -w option is used.

The following example is the regular grep where it is searching for “is”. When you search for “is”, without any option it will show out “is”, “his”, “this” and everything which has the substring “is”.

```
$ grep -i "is" demo_file
THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.
Two lines above this line is empty.
And this is the last line.
```

The following example is the WORD grep where it is searching only for the word “is”

```
$ grep -iw “is” demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
Two lines above this line is empty.
And this is the last line.
```

1.8.6. Searching in all files recursively

When you want to search in all the files under the current directory and its sub directory ‘–r’ option is the one which you need to use. The following example will look for the string “ramesh” in all the files in the current directory and all its subdirectory.

```
$ grep -r "ramesh" *
```

1.8.7. Invert match

If you want to display the lines which does not matches the given string/pattern, use the option -v as shown below. This example will display all the lines that did not match the word “Two”.

```
$ grep -v Two demo_file
THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.
And this is the last line.
```

Displaying the lines which does not matches the entire given pattern.

Syntax:

grep -v -e pattern -e pattern

For example, the file **file1** has the following content

Apple

```
Banana
Cauliflower
Grapes
Orange
```

```
$ grep -v -e “apple” -e “cauliflower” file1
Banana
Grapes
Orange
```

1.8.8. Counting the number of matches

Count the number of lines matched in the given pattern/string, then use the option -c.

Syntax:

grep -c pattern filename

1.8.9. Displaying only the file names which matches the given pattern

The -l option is used to display only the file names which matched the given pattern. When you give multiple files to the grep as input, it displays the names of file which contains the text that matches the pattern, will be very handy when you try to find some notes in your whole directory structure.

```
$ grep -l this demo_*
demo_file
demo_file1
```

1.8.10. Showing line number while displaying the output

To show the line number of file with the line matched, -n option is used.

Syntax:

grep -n pattern filename

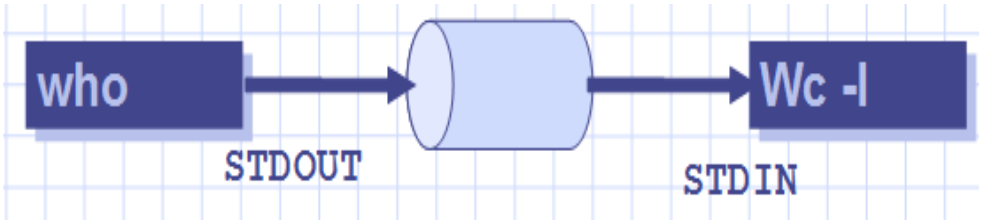
Example:

grep -n "this" demo_file

```
2: this line is the 1st lower case line in this file.
6: Two lines above this line is empty.
```

1.9. Pipe

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. In shell the symbol '|' is used to represent pipe.



Example:

```
$ who | wc –l
```

Purpose of Pipes

Using pipe you can construct powerful unix command lines by combining basic unix commands. UNIX commands are powerful; however by using pipe you can combine them together, to accomplish complex tasks with ease.

Through the standard output of one command (the command to the left of the pipe) gets sent as standard input to another command (the command to the right of the pipe). Pipe functions in a similar manner like the output redirection in UNIX (using > symbol to redirect the standard output of a command to a file. However, the pipe is different because it is used to pass the output of a command to another command, not a file.

Example:

```
$ cat apple.txt
core
worm seed
jewel
```

```
$ cat apple.txt | wc
     3      4     21
```

In this example, the contents of the file apple.txt are sent through pipe to wc (word count)

command. The wc command then does its job and counts the lines, words, and characters in the file.

You can combine many commands with pipes on a single command line. Here's an example where the characters, words, and lines of the file apple.txt is sent to wc and then the output of wc mailed to nobody@december.com with the subject line "The count."

```
$ cat apple.txt | wc | mail -s "The count" nobody@december.com
```