Filters

A filter is a program that takes its input from the standard input file, process(or filters) it, and send its output to the standard output file. Unix has a rich set of filters that can be used to work on data in an effective way. Some examples of filters are grep,wc , tr,cut and sort

head Command

Head prints the first N number of data of the given input. By default, it prints first 10 lines of each given file.

**Syntax and Options**

head [OPTIONS]… [FILE]…

|  |  |
| --- | --- |
| **Option** | **Option Description** |
| -c | to print N bytes from each input file. |
| -n | to print N lines from each input file. |

Example

**(1)Display default first 10 lines of given file**

**$head myfile.txt**

This command prints the first 10 lines of myfile.txt

**(2) Print the first N number of lines**

To view the first N number of lines, pass the file name as an argument with -n option as shown below. This will display the first 5 lines of myfile.txt

**$ head -n 5 myfile.txt**

Or

**$ head -5 myfile.txt**

**(3) Print the N number of bytes or characters**

You can use the -c option to print the N number of bytes from the initial part of file.

**$ head -c 5 myfile.txt**

**(4) Print all but not the last N lines**

By placing ‘-’ in front of the number with -n option, it prints all the lines of each file but not the last N lines.

**$ head -n -5 myfile.txt**

**(5)Passing Output of Other command to Head Input**

You may pass the output of other commands to the head command via pipe as shown below,

**$ls –l | head –n 5**

**$ls | head | sort -r >> file1**

tail command

**tail command is used to display the last or bottom part of the file. By default it displays last 10 lines of a file.**  
**Syntax and Options**

 tail [options] filename

|  |  |
| --- | --- |
| **Option** | **Option Description** |
| -c | Prints the last N bytes of file; With leading +, prints the characters from the N byte in the file. |
| -n | Prints last N lines; With leading + prints lines from the Nth line in the file. |

Example

**(1)Display default first 10 lines of given file**

By default, the tail command **prints the last 10 lines from the file**.

**$tail myfile.txt**

**(2) Print the last N number of lines from the file**

To view the last N number of lines, pass the file name as an argument with -n option as shown below. This will display the last 5 lines of myfile.txt

**$ tail -n 5 myfile.txt**

**Or**

**$ tail -5 myfile.txt**

**(3) Print the last N bytes from the file**

The following example prints the last 8 bytes from the file.

**$tail -c8 myfile.txt**

**(4) Print lines from the Nth line**

The following example prints lines from the 7th line.

**$tail -n+7 myfile.txt**

Cut command

Cut command in unix is used to select sections of text from each line of files. You can use the cut command to select fields or columns from a line by specifying a delimiter or you can select a portion of text by specifying the range or characters. Basically the cut command slices a line and extracts the text.

**SYNTAX:**  
cut [options]

|  |  |
| --- | --- |
| **Option** | **Option Description** |
| -c | specifies a range of characters which will be returned, e.g. cut -c1-8 would return the first 8 characters of a line |
| -f | Specifies a field list, separated by a [delimiter](http://en.wikipedia.org/wiki/Delimiter) |
| -d | the -d option is the field delimiter for use in conjunction with the -f option; the default delimiter is *tab* |

**Example**

**(1) The cut command can be used to print characters in a line by  
 specifying the position of the characters. To print the characters   
 in a line, use the -c option in cut command**

**$cut -c4 empfile.txt**

**(2) You can print more than one character at a time by specifying the character positions in a comma separated list as shown in the below example**

**$cut -c4,6 empfile.txt**

This command prints the fourth and sixth character in each line.

**(3) You can print a range of characters in a line by specifying the start and end position of the characters.**

**$cut -c4-7 empfile.txt**

The above cut command prints the characters from fourth position to the seventh position in each

**(4) You can use the cut command to extract the fields in a file using a delimiter. The -d option in cut command can be used to specify the delimiter and -f option is used to specify the field position.**

**$cut -d' ' -f2 empfile.txt**

This command prints the second field in each line by treating the space as delimiter.

**(5) You can print more than one field by specifying the position of the fields in a comma delimited list.**

**$cut -d' ' -f2,3 empfile.txt**

The above command prints the second and third field in each line.

**(6) You can print a range of fields by specifying the start and end position.**

**$cut -d' ' -f1-3 empfile.txt**

The above command prints the first, second and third fields.

**$who | cut -d' ' -f1**

paste Command

Paste command is one of the useful commands in unix or linux operating system. The paste command merges the lines from multiple files. The paste command sequentially writes the corresponding lines from each file separated by a TAB delimiter on the unix terminal.   
  
 paste command is used to paste the content from one file to another file. It is also used to set column format for each line.

The syntax of the paste command is

paste [options] files-list

The options of paste command are:

-d : Specify of a list of delimiters.

-s : Paste one file at a time instead of in parallel

**Paste Command Examples**:  
  
Create the following three files

**$ cat file1**

**Unix**

**Linux**

**Windows**

**$ cat file2**

**Dedicated server**

**Virtual server**

**Normal server**

**1. Merging files in parallel**  
By default, the paste command merges the files in parallel. The paste command writes corresponding lines from the files as a tab delimited on the terminal.

**$ paste file1 file2**

**Unix Dedicated server**

**Linux Virtual server**

**Windows Normal server**

**2. Specifying the delimiter**  
Paste command uses the tab delimiter by default for merging the files. You can change the delimiter to any other character by using the -d option.

**$ paste -d"|" file1 file2**

**Unix|Dedicated server**

**Linux|Virtual server**

**Windows|Normal server**

In the above example, pipe delimiter is specified  
  
**3. Merging files in sequentially.**  
You can merge the files in sequentially using the -s option. The paste command reads each file in sequentially. It reads all the lines from a single file and merges all these lines into a single line.

**$ paste -s file1 file2**

**Unix Linux Windows**

**Dedicated server Virtual server Normal server**

**(4)The following example shows how to specify a delimiter for sequential merging of files:**

**$ paste -s -d"," file1 file2**

**Unix,Linux,Windows**

**Dedicated server,Virtual server, Normal server**

**(5)Paste the contents of file1 and file2 to file3**

**$ paste -d"|" file1 file2 >file3**

**But contents of the file 3 are deleted.**

tr command

tr is an UNIX utility for translating, or deleting, or squeezing repeated characters. It will read from STDIN and write to STDOUT.

tr stands for translate.

**Syntax**

The syntax of tr command is:

$ tr [OPTION] SET1 [SET2]

Examples

**1. Convert lower case to upper case**

**$cat empfile | tr “[a-z]” “[A-Z]”**

**or**

**$cat empfile | tr “[:lower:]” “[:upper:]”**

**2. Translate white-space to tabs**

The following command will translate all the white-space to tabs

**$ echo "This is for testing" | tr [:space:] '\t'**

**This is for testing**

**3. Translate braces into parenthesis**

You can also translate from and to a file. In this example we will translate braces in a file with parenthesis.

**$ tr '{}' '()' < myfile.txt > newfile.txt**

The above command will read each character from “myfile.txt”, translate if it is a brace, and write the output in “newfile.txt”.

**3. Squeeze repetition of characters using -s**

You can convert multiple continuous spaces with a single space

**$ echo "This is testing" | tr -s [:space:] ' '**

**This is testing**

**4. Delete specified characters using -d option**

tr can also be used to remove particular characters using -d option.

**$ echo "This is for testing" | tr -d 't'**

**his is for esing**

To remove all the digits from the string, use

**$ echo "my username is 432234" | tr -d [:digit:]**

**my username is**

**5. Complement the sets using -c option**

You can complement the SET1 using -c option. For example, to remove all characters except digits, you can use the following.

**$ echo "my username is 432234" | tr -cd [:digit:]**

**432234**

**Or**

**$ echo “unix” | tr –c “u” “a”**

**Uaaa**

sort command

**sort** command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ascii. Using options in sort command, it can also be used to sort numerically

The syntax of sort command is:

sort [options] filename

The options are:

-f : Uses case insensitive sorting.

-n : Uses numeric sorting

-r : Reverse order sorting

-k : Sorts file based on the data in the  
 specified field positions.

-u : Suppresses duplicate lines

-o : Write result to FILE instead of standard  
 output duplicate lines

Example

 Let us consider a file with the following contents:

**$ cat > myfile**

Unix

Linux

Solaris

AIX

Linux

Linux

HPUX

**(1) sort simply sorts the file in alphabetical order**

**$** **sort myfile**

AIX

HPUX

Linux

Linux

Linux

Solaris

Unix

or

**$** **sort < myfile**

AIX

HPUX

Linux

Linux

Linux

Solaris

Unix

**$** **sort myfile >filedata**

**Sorts and inputs into a new file**

**(2) sort removes the duplicates using the -u option**

**$ sort -u myfile**

AIX

HPUX

Linux

Solaris

Unix

The duplicate 'Linux' record got removed. '-u' option removes all the duplicate records in the file.

**(3) To sort a file numerically used –n option**

 Let us consider a file with numbers:

**$ cat >numfile**

20

19

5

49

200

**$ sort -n numfile**

5

19

20

49

200

**(4) sort file in reverse order used –r option**

**$** **sort –r myfile**

Unix

Solaris

Linux

Linux

Linux

HPUX

AIX

**(5) sort file numerically in reverse order:**

**$ sort -nr numfile**

200

49

20

19

5

**sort can sort multiple files**

**$ sort -n numfile1 numfile2**

**(6) Use -o option to save the sorted output to a file.**

**$sort myfile –o datafile.txt**

**(7) sort on nth field**

 Let us consider a file with the following contents:

**$ cat >empdata**

ajay manager 5000

sunil clerk 4000

amit manager 6000

deepak peon 4500

chetan director 9000

rajesh peon 3000

**$sort –k2 empdata**

sunil clerk 4000

chetan director 9000

ajay manager 5000

amit manager 6000

rajesh peon 3000

deepak peon 4500

**It will sort according to 2 field**