**Find: search files, and folder**

**AWK : for data extraction | find content**

**Sample Commands**

**Example:**  
Consider the following text file as the input file for all cases below.

$cat > employee.txt

ajay manager account 45000

sunil clerk account 25000

varun manager sales 50000

amit manager account 47000

tarun peon sales 15000

deepak clerk sales 23000

sunil peon sales 13000

satvik director purchase 80000

**1. Default behavior of Awk :** By default Awk prints every line of data from the specified file.

$ awk '{print}' employee.txt

**Output:**

ajay manager account 45000

sunil clerk account 25000

varun manager sales 50000

amit manager account 47000

tarun peon sales 15000

deepak clerk sales 23000

sunil peon sales 13000

satvik director purchase 80000

In the above example, no pattern is given. So the actions are applicable to all the lines. Action print without any argument prints the whole line by default, so it prints all the lines of the file without failure.

**2. Print the lines which matches with the given pattern.**

$ awk '/manager/ {print}' employee.txt

**Output:**

ajay manager account 45000

varun manager sales 50000

amit manager account 47000

In the above example, the awk command prints all the line which matches with the ‘manager’.

**3. Spliting a Line Into Fields :** For each record i.e line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

$ awk '{print $1,$4}' employee.txt

$ awk '/manager/ {print $1,$4}' employee.txt

**Output:**

ajay 45000

sunil 25000

varun 50000

amit 47000

tarun 15000

deepak 23000

sunil 13000

satvik 80000

In the above example, $1 and $4 represents Name and Salary fields respectively.

**Built In Variables In Awk**

Awk’s built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

**NR:** NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file.

**NF:** NF command keeps a count of the number of fields within the current input record.

**FS:** FS command contains the field separator character which is used to divide fields on the input line. The default is “white space”, meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator.

**RS:** RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.

**OFS:** OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.

**ORS:** ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

**Examples:**

**Use of NR built-in variables (Display Line Number)**

$ awk '{print NR,$0}' employee.txt

**Output:**

1 ajay manager account 45000

2 sunil clerk account 25000

3 varun manager sales 50000

4 amit manager account 47000

5 tarun peon sales 15000

6 deepak clerk sales 23000

7 sunil peon sales 13000

8 satvik director purchase 80000

In the above example, the awk command with NR prints all the lines along with the line number.

**Use of NF built-in variables (Display Last Field)**

$ awk '{print $1,$NF}' employee.txt

**Output:**

ajay 45000

sunil 25000

varun 50000

amit 47000

tarun 15000

deepak 23000

sunil 13000

satvik 80000

In the above example $1 represents Name and $NF represents Salary. We can get the Salary using $NF , where $NF represents last field.

**Another use of NR built-in variables (Display Line From 3 to 6)**

$ awk 'NR==3, NR==6 {print NR,$0}' employee.txt

**Output:**

3 varun manager sales 50000

4 amit manager account 47000

5 tarun peon sales 15000

6 deepak clerk sales 23000

**More Examples**

**For the given text file:**

$cat > geeksforgeeks.txt

A B C

Tarun A12 1

Man B6 2

Praveen M42 3

**1) To print the first item along with the row number(NR) separated with ” – “ from each line in geeksforgeeks.txt:**

$ awk '{print NR "- " $1 }' geeksforgeeks.txt

1 - Tarun

2 – Manav

3 - Praveen

**2) To return the second row/item from geeksforgeeks.txt:**

$ awk '{print $2}' geeksforgeeks.txt

A12

B6

M42

**3) To print any non empty line if present**

$ awk 'NF > 0' geeksforgeeks.txt

0

**4) To find the length of the longest line present in the file:**

$ awk '{ if (length($0) > max) max = length($0) } END { print max }' geeksforgeeks.txt

13

**5) To count the lines in a file:**

$ awk 'END { print NR }' geeksforgeeks.txt

3

**6) Printing lines with more than 10 characters:**

$ awk 'length($0) >10' geeksforgeeks.txt

Tarun A12 1

Praveen M42 3

**7) To find/check for any string in any column:**

$ awk '{ if($3 == "B6") print $0;}' geeksforgeeks.txt

**8) To print the squares of first numbers from 1 to n say 6:**

$ awk 'BEGIN { for(i=1;i<=5;i++) print "square of", i, "is",i\*i; }'

square of 1 is 1

square of 2 is 4

square of 3 is 9

square of 4 is 16

square of 5 is 25

square of 6 is 36

**Sed Command in Linux/Unix with examples**

SED command in UNIX is stands for stream editor and it can perform lot’s of function on file like, searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace. By using SED you can edit files even without opening it, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it.

* SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution).
* SED command in unix supports regular expression which allows it perform complex pattern matching.

**Syntax:**

**sed OPTIONS... [SCRIPT] [INPUTFILE...]**

**Example:**  
Consider the below text file as an input.

**$cat > geekfile.txt**

unix is great os. unix is opensource. unix is free os.

learn operating system.

unix linux which one you choose.

unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

**Sample Commands**

1. **Replacing or substituting string :** Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.
2. **$sed 's/unix/linux/' geekfile.txt**

**Output :**

linux is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

Here the “s” specifies the substitution operation. The “/” are delimiters. The “unix” is the search pattern and the “linux” is the replacement string.

By default, the sed command replaces the first occurrence of the pattern in each line and it won’t replace the second, third…occurrence in the line.

1. **Replacing the nth occurrence of a pattern in a line :** Use the /1, /2 etc flags to replace the first, second occurrence of a pattern in a line. The below command replaces the second occurrence of the word “unix” with “linux” in a line.
2. **$sed 's/unix/linux/2' geekfile.txt**

**Output:**

unix is great os. linux is opensource. unix is free os.

learn operating system.

unix linux which one you choose.

unix is easy to learn.linux is a multiuser os.Learn unix .unix is a powerful.

1. **Replacing all the occurrence of the pattern in a line :** The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.
2. **$sed 's/unix/linux/g' geekfile.txt**

**Output :**

linux is great os. linux is opensource. linux is free os.

learn operating system.

linux linux which one you choose.

linux is easy to learn.linux is a multiuser os.Learn linux .linux is a powerful.

1. **Replacing from nth occurrence to all occurrences in a line :** Use the combination of /1, /2 etc and /g to replace all the patterns from the nth occurrence of a pattern in a line. The following sed command replaces the third, fourth, fifth… “unix” word with “linux” word in a line.
2. **$sed 's/unix/linux/3g' geekfile.txt**

**Output:**

unix is great os. unix is opensource. linux is free os.

learn operating system.

unix linux which one you choose.

unix is easy to learn.unix is a multiuser os.Learn linux .linux is a powerful.

1. **Parenthesize first character of each word :** This sed example prints the first character of every word in paranthesis.
2. **$ echo "Welcome To The Geek Stuff" | sed 's/\(\b[A-Z]\)/\(\1\)/g'**

Output:

(W)elcome (T)o (T)he (G)eek (S)tuff

1. **Replacing string on a specific line number :** You can restrict the sed command to replace the string on a specific line number. An example is
2. **$sed '3 s/unix/linux/' geekfile.txt**

**Output:**

unix is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

The above sed command replaces the string only on the third line.

1. **Duplicating the replaced line with /p flag :** The /p print flag prints the replaced line twice on the terminal. If a line does not have the search pattern and is not replaced, then the /p prints that line only once.
2. **$sed 's/unix/linux/p' geekfile.txt**

**Output:**

linux is great os. unix is opensource. unix is free os.

linux is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

linux linux which one you choose.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

1. **Printing only the replaced lines :** Use the -n option along with the /p print flag to display only the replaced lines. Here the -n option suppresses the duplicate rows generated by the /p flag and prints the replaced lines only one time.
2. **$sed -n 's/unix/linux/p' geekfile.txt**

**Output:**

linux is great os. unix is opensource. unix is free os.

linux linux which one you choose.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

If you use -n alone without /p, then the sed does not print anything.

1. **Replacing string on a range of lines :** You can specify a range of line numbers to the sed command for replacing a string.
2. **$sed '1,3 s/unix/linux/' geekfile.txt**

**Output:**

linux is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

Here the sed command replaces the lines with range from 1 to 3. Another example is

**$sed '2,$ s/unix/linux/' geekfile.txt**

**Output:**

unix is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful

Here $ indicates the last line in the file. So the sed command replaces the text from second line to last line in the file.

1. **Deleting lines from a particular file :** SED command can also be used for deleting lines from a particular file. SED command is used for performing deletion operation without even opening the file  
   Examples:  
   1. To Delete a particular line say n in this example
2. Syntax:
3. $ sed 'nd' filename.txt
4. Example:
5. $ sed '5d' filename.txt

2. To Delete a last line

Syntax:

$ sed '$d' filename.txt

3. To Delete line from range x to y

Syntax:

$ sed 'x,yd' filename.txt

Example:

$ sed '3,6d' filename.txt

5. To Delete from nth to last line

Syntax:

$ sed 'nth,$d' filename.txt

Example:

$ sed '12,$d' filename.txt

6. To Delete pattern matching line

Syntax:

$ sed '/pattern/d' filename.txt

Example:

$ sed '/abc/d' filename.txt

1. Viewing a range of lines of a document

# sed -n '5,10p' myfile.txt

### 2. Viewing the entire file except a given range

On the other hand, it’s possible that you want to print the entire file except a certain range. To exclude lines **20** through **35** from **myfile.txt**, do:

# sed '20,35d' myfile.txt

### 4. Replacing words or characters (basic substitution)

To replace every instance of the word version with story in **myfile.txt**, do:

# sed 's/version/story/g' myfile.txt

### 5. Replacing words or characters inside a range

If you’re interested in replacing words only within a line range (**30** through **40**, for example), you can do:

# sed '30,40 s/version/story/g' myfile.txt

### 6. Using regular expressions (advanced substitution) – I

Sometimes configuration files are loaded with comments. While this is certainly useful, it may be helpful to display only the configuration directives sometimes if you want to view them all at a glance.

To remove empty lines or those beginning with # from the Apache configuration file, do:

# sed '/^#\|^$\| \*#/d' httpd.conf

### 7. Using regular expressions (advanced substitution) – II

To replace a word beginning with uppercase or lowercase with another word, we can also use sed. To illustrate, let’s replace the word **zip** or **Zip** with rar in **myfile.txt**:

# sed 's/[Zz]ip/rar/g' myfile.txt

### 8. Viewing lines containing with a given pattern

Another use of **sed** consists in printing the lines from a file that match a given regular expression. For example, we may be interested in viewing the authorization and authentication activities that took place on **July 2**, as per the **/var/log/secure** log in a **CentOS 7** server.

In this case, the pattern to search for is **Jul 2** at the beginning of each line:

# sed -n '/^Jul 1/ p' /var/log/secure

### 9. Inserting spaces in files

With **sed**, we can also insert spaces (blank lines) for each non-empty line in a file. To insert one blank line every other line in **LICENSE**, a plain text file, do:

# sed G myfile.txt

To insert two blank lines, do:

# sed 'G;G' myfile.txt

Rplae value infile

sed -i 's/original/new/g' file.txt

Explanation:

* sed = Stream EDitor
* -i = in-place (i.e. save back to the original file)
* The command string:
  + s = the substitute command
  + original = a regular expression describing the word to replace (or just the word itself)
  + new = the text to replace it with
  + g = global (i.e. replace all and not just the first occurrence)
* file.txt = the file name

1. Assuming that you want to find all files in the current directory with .sh and .txt file extensions, you can do this by running the command below:

# find . -type f \( -name "\*.sh" -o -name "\*.txt" \)

2. To find three filenames with .sh, .txt and .c extensions, issues the command below:

# find . -type f \( -name "\*.sh" -o -name "\*.txt" -o -name "\*.c" \)