Searching for pattern using grep (grep)

grep searches file for a pattern and display matching or non-matching lines. It search for a pattern and display line containing the pattern, the line numbers or filenames where the pattern occurs.

Syntax:

grep options pattern filename(s)

```
$ grep "Surat" student.lst
1111    Amit B. Shah          01/01/1980  Surat    bca     9429123423
1114    Samir P. Chowdhari    12/02/1982  Surat    bcom    8461626364
2111    Bimal R. Surati       10/08/1990  Navsari  ba      9223423456
$
```

If the pattern does not found it simple return the prompt.

```
$ grep "Anand" student.lst
$ _
```

grep is also used with multiple file names, it display the filename with the output.

```
$ grep "Ahmedabad" student.lst Student1.lst
Student.lst:1115    Vimal K. Patel      18/03/1984  Ahmedabad  bca  9526126126
Student1.lst:3124   Rita K. Makhwana    19/09/1976  Ahmedabad  ba   9898212123
Student.lst:1115    Vimal K. Patel      18/03/1984  Ahmedabad  bca  9526126126
Student1.lst:3124   Rita K. Makhwana    19/09/1976  Ahmedabad  ba   9898212123
$
```

grep options

Ignoring Case (-i)

When you want to search name but don't know case, use the –i (ignore case) option. This option ignores case for pattern matching.

```
$ grep –i 'NAVSARI' student.lst
2111 | Bimal R. Surati     | 10/08/1990 | Navsari  | ba  | 9223423456
$
```

Deleting lines (-v)

The –v option select all lines except those containing pattern.

```
$ $ grep -v "Bharuch" student.lst>student2.lst

$
```

Displaying Line Numbers (-n)

The –n option display line numbers with records containing the pattern.

```
$ grep -n "Ahmedabad" student.lst
5:1115  | Vimal K. Patel    | 18/03/1984 | Ahmedabad | bca | 9526126126
12:3124 | Rita K. Makhwana  | 19/09/1976 | Ahmedabad | ba  | 9898212123

$
```

Counting Lines Containing Pattern (-c)

The –c option counts the number of lines containing the pattern and display it.

```
$ grep –c "Ahmedabad" student.lst
2
```

```
$
```

The –c option can also used with multiple filename.

```
$ grep -c "Ahmedabad" student.lst Student1.lst
Student.lst:2
Student1.lst:2

$
```

Display name of file containing pattern (-l)

The –l option displays only the name of files containing the pattern:

```
$ grep -l "Bharuch" *.lst
student.lst
student1.lst

$
```

Matching Multiple Patterns (-e)

The –e option is used to match the multiple patterns. For example if we want to match the three Makwanas with different spells then:

```
$ grep –e "Makwana" –e "Makvana" –e "Makhwana" student.lst
1113 | Sunil C. Makwana  | 11/03/1981 | Vapi      | Bca | 9925421213
3122 | Nilesh K. Makvana | 25/03/1976 | Bharuch   | Bca | 9712323456
3124 | Rita K. Makhwana  | 19/09/1976 | Ahmedabad | Ba  | 9898212123

$
```

Taking Patterns from a File (-f)

We can put the pattern in separate file. The –f option take pattern from the first file and search the pattern in the second file and display the result.

```
$ cat > pat.lst
Makwana
Makvana
Makhwana
```

```
$ grep –f pat.lst student.lst
1113 | Sunil C. Makwana  | 11/03/1981 | Vapi      | Bca | 9925421213
3122 | Nilesh K. Makvana | 25/03/1976 | Bharuch   | Bca | 9712323456
3124 | Rita K. Makhwana  | 19/09/1976 | Ahmedabad | Ba  | 9898212123
```

Basic Regular Expression

grep uses an expression of a different type to match a group of similar patterns. It uses an elaborate metacharacter set, and perform amazing matches. The expression which uses these metacharacter set is called Basic Regular Expression. Regular expressions are interpreted by the command and not by the shell.

Symbol or expression	Description
*	Zero or more occurrence of previous character
g*	Nothing or g, gg, ggg, etc.
.	A single character
.*	Nothing or any number of characters
[pqr]	A single character p,q or r
[a-d]	A single character a, b, c or d

[1-3]	A digit eighter 1, 2 or 3
[^pqr]	A single character which is not p, q or r.
[^a-zA-Z]	A non alphabetic character
^pat	Pattern *pat* at beginning of line
pat$	Pattern *pat* at end of line
^Hello$	Hello is the only word in line
^$	Lines containing nothing

The Dot (.)

The **.** metacharacter is used matches a single character.

1..

The above pattern matches three character patterns beginning with a 1.

The Regular Expression .*

The **.*** signifies zero or more characters.

Example suppose you want to search Name Anil A. Joshi. But you are not sure that whether it is A. A. Joshi or Anil A. Joshi in file. We can serach this name using .* in search string.

```
$ grep "A.*Joshi" student.lst
2112 | Anil A. Joshi       | 15/02/1985 | Valsad      | bba  | 9712312342
$
```

Specifying Pattern Locations (^ and $)

The two metacharacters ^ and $ can match pattern at the beginning or end of a line.

^ (caret) – For matching at the beginning of the line.

$ - For matching at the end of a line.

Ex: Display the record of students whose number starts with “2”.

```
$ grep “^2” student.lst

2223 | Ronak S. Surti      | 14/08/1970 | Surat     | bba  | 9595223231

2234 | Jinal B. Chaudhri   | 17/07/1977 | Bharuch   | bca  | 9427194271
```

Display record of students whose mobile number start with 8.

```
$ grep “8………$” student.lst

1114 | Samir P. Chowdhari  | 12/02/1982 | Surat     | bcom | 8461626364

$
```

Find the students whose number not start with 1.

```
$ grep “^[^1]” student.lst

2111 | Bimal R. Surati     | 10/08/1990 | Navsari   | ba   | 9223423456
2112 | Anil A. Joshi       | 15/02/1985 | Valsad    | bba  | 9712312342
2223 | Ronak S. Surti      | 14/08/1970 | Surat     | bba  | 9595223231
2234 | Jinal B. Chaudhri   | 17/07/1977 | Bharuch   | bca  | 9427194271
3122 | Nilesh K. Makvana   | 25/03/1976 | Bharuch   | bca  | 9712323456
3123 | Samira S. Chodhari  | 17/07/1977 | Bharuch   | bcom | 9712323456
3124 | Rita K. Makhwana    | 19/09/1976 | Ahmedabad | Ba   | 9898212123
```

List all directories in current path.

```
$ Ls –l | grep “^d”
```

Display records where rollno start with 1 or 2.

```
$ grep "^[12]" student.lst
1111 | Amit B. Shah       | 01/01/1980 | Surat     | bca  | 9429123423
1112 | Ramesh S. Chaudhari| 20/02/1982 | Baroda    | bba  | 9926412126
1113 | Sunil C. Makwana   | 11/03/1981 | Vapi      | bca  | 9925421213
1114 | Samir P. Chowdhari | 12/02/1982 | Surat     | bcom | 8461626364
1115 | Vimal K. Patel     | 18/03/1984 | Ahmedabad | bca  | 9526126126
2111 | Bimal R. Surati    | 10/08/1990 | Navsari   | ba   | 9223423456
2112 | Anil A. Joshi      | 15/02/1985 | Valsad    | bba  | 9712312342
2223 | Ronak S. Surti     | 14/08/1970 | Surat     | bba  | 9595223231
2234 | Jinal B. Chaudhri  | 17/07/1977 | Bharuch   | bca  | 9427194271
```

EXTENDED REGULAR EXPRESSION (ERE)

To match dissimilar patterns with a single expression Extended regular expressions (ERE) is used. It uses some additional character. To use Extended regular expression in grep is possible using –E option.

Special character + and ?

The ERE include two special characters + and ?.

+	Matching one or more occurrence of previous character.
?	Matches zero or one occurrence of previous character.

It means **a**+ matching **a**, **aa**, **aaa** etc…. while **b**? matches a single instance of **b** or nothing.

$ grep –E "Sura?ti" student.lst

2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**ba**	**9223423456**
2223	**Ronak S. Surti**	**14/08/1970**	**Surat**	**bba**	**9595223231**

Matching Multiple Patterns (| ,(|))

The | is the delimiter of multiple patterns. For Example we can find both Vimal and Bimal from the file and without using –e option twice.

$ grep –E 'Vimal|Bimal' student.lst

1115	**Vimal K. Patel**	**18/03/1984**	**Ahmedabad**	**bca**	**9526126126**
2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**ba**	**9223423456**

The same output with more compact pattern can be done through **|** inside the **()**.

$grep –E '(Vi|Bi)mal' student.lst

1115	**Vimal K. Patel**	**18/03/1984**	**Ahmedabad**	**bca**	**9526126126**
2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**ba**	**9223423456**

The Stream Editor (sed)

sed is a tool that is design for multipurpose. sed uses instructions to act on text. An instruction combines an address for selecting lines, with an action to be taken on them.

Syntax :

sed options 'address action' file(s)

The address and action are enclosed within single quotes. Addressing in sed is done in two ways:

- By one or two line numbers (like 2,5)
- By specifying a / enclosed pattern which occurs in a line.(like /Navsari/).

Line addressing

Quit (q)

The q is quit after display specified lines. For example 5q display first 5 lines and quit. It is similar to head –n 5 .

```
$ sed '5q' student.lst
1111 | Amit B. Shah         | 01/01/1980 | Surat     | bca  | 9429123423
1112 | Ramesh S. Chaudhari  | 20/02/1982 | Baroda    | bba  | 9926412126
1113 | Sunil C. Makwana     | 11/03/1981 | Vapi      | bca  | 9925421213
1114 | Samir P. Chowdhari   | 12/02/1982 | Surat     | bcom | 8461626364
1115 | Vimal K. Patel       | 18/03/1984 | Ahmedabad | bca  | 9526126126
```

Print (p)

p command is used in sed to display lines. But it behaves in strange manner. It display selected line as well as all lines. So we must give –n option whenever we use **p**.

To display line 1 to 3 from student.lst

```
$ sed –n '1,3p' student.lst
1111 | Amit B. Shah         | 01/01/1980 | Surat     | bca  | 9429123423
1112 | Ramesh S. Chaudhari  | 20/02/1982 | Baroda    | bba  | 9926412126
1113 | Sunil C. Makwana     | 11/03/1981 | Vapi      | bca  | 9925421213
```

To display last line of the student.lst file.

$ sed –n '$p' student.lst

3124	**Rita K. Makhwana**	**19/09/1976**	**Ahmedabad**	**ba**	**9898212123**

Selected line from anyware

sed can select group of lines from any where in the file

To select line 8 to 10 from student.lst

$ sed –n '8,10p' student.lst

2223	**Ronak S. Surti**	**14/08/1970**	**Surat**	**bba**	**9595223231**
2234	**Jinal B. Chaudhri**	**17/07/1977**	**Bharuch**	**bca**	**9427194271**
3122	**Nilesh K. Makvana**	**25/03/1976**	**Bharuch**	**bca**	**9712323456**

Select Multiple Group of Lines

It is also possible with sed to select multiple group of lines.

Display line 1 to 3, 5 to 8 and last line from student.lst

```
$ sed –n '1,3p
  5,8p
  $p' student.lst
```

1111	**Amit B. Shah**	**01/01/1980**	**Surat**	**bca**	**9429123423**
1112	**Ramesh S. Chaudhari**	**20/02/1982**	**Baroda**	**bba**	**9926412126**
1113	**Sunil C. Makwana**	**11/03/1981**	**Vapi**	**bca**	**9925421213**

1115	**Vimal K. Patel**	**18/03/1984**	**Ahmedabad**	**bca**	**9526126126**
2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**ba**	**9223423456**
2112	**Anil A. Joshi**	**15/02/1985**	**Valsad**	**bba**	**9712312342**
2223	**Ronak S. Surti**	**14/08/1970**	**Surat**	**bba**	**9595223231**
3124	**Rita K. Makhwana**	**19/09/1976**	**Ahmedabad**	**ba**	**9898212123**

Negative the action (!)

sed also has a negative operation **!** which can be used with any action. To display first three lines is same as not to display line 4 to end.

```
$ sed –n '4,$!p' student.lst
```

1111	**Amit B. Shah**	**01/01/1980**	**Surat**	**bca**	**9429123423**
1112	**Ramesh S. Chaudhari**	**20/02/1982**	**Baroda**	**bba**	**9926412126**
1113	**Sunil C. Makwana**	**11/03/1981**	**Vapi**	**bca**	**9925421213**

Using multiple instructions (-e and –f)

The –e option allows you to enter multiple instructions in one command, each instruction is preceded by –e option.

$ sed –n –e '1,3p' –e '5,7p' –e '9,10p' student.lst

1111	**Amit B. Shah**	**01/01/1980**	**Surat**	**bca**	**9429123423**
1112	**Ramesh S. Chaudhari**	**20/02/1982**	**Baroda**	**bba**	**9926412126**
1113	**Sunil C. Makwana**	**11/03/1981**	**Vapi**	**bca**	**9925421213**
1115	**Vimal K. Patel**	**18/03/1984**	**Ahmedabad**	**bca**	**9526126126**
2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**ba**	**9223423456**
2112	**Anil A. Joshi**	**15/02/1985**	**Valsad**	**bba**	**9712312342**
2234	**Jinal B. Chaudhri**	**17/07/1977**	**Bharuch**	**bca**	**9427194271**
3122	**Nilesh K. Makvana**	**25/03/1976**	**Bharuch**	**bca**	**9712323456**

When there are many instructions to use in sed then stored instructions in file with each instruction on separate line. The above example can be stored in a file with each instruction on separate line.

```
$ cat > instr1.txt
1,3p
5,7p
9,10p
[Ctrl-d]
```

Now use the –f option to direct sed to take its instructions from the file using following command:

```
$ sed –n –f instr1.txt student.lst
1111 | Amit B. Shah       | 01/01/1980 | Surat     | bca | 9429123423
1112 | Ramesh S. Chaudhari| 20/02/1982 | Baroda    | bba | 9926412126
1113 | Sunil C. Makwana   | 11/03/1981 | Vapi      | bca | 9925421213
1115 | Vimal K. Patel     | 18/03/1984 | Ahmedabad | bca | 9526126126
2111 | Bimal R. Surati    | 10/08/1990 | Navsari   | ba  | 9223423456
2112 | Anil A. Joshi      | 15/02/1985 | Valsad    | bba | 9712312342
2234 | Jinal B. Chaudhri  | 17/07/1977 | Bharuch   | bca | 9427194271
3122 | Nilesh K. Makvana  | 25/03/1976 | Bharuch   | bca | 9712323456
```

You can use –f option with multiple files.

```
$ Cat > instr2.txt
2p
$p
[Ctrl-d]
```

$ sed –n –f instr1.txt –f instr2.txt student.lst

You can also combine –e and –f options many times in one command.

$ sed –n –e '/Patel/p' -f instr1.txt student.lst

Context Addressing

The second form of addressing is called context addressing. It lets you specify one or more patterns to locate lines. The pattern must be enclosed by a /. When you specify a pattern, it select all lines containing that pattern and displayed.

$ sed –n '/Surati/p' student.lst

2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**Ba**	**9223423456**

You can specify comma separated many context addresses to select a group of lines.

$ sed –n '/Patel/,/Surati/p' student.lst

1115	**Vimal K. Patel**	**18/03/1984**	**Ahmedabad**	**bca**	**9526126126**
2111	**Bimal R. Surati**	**10/08/1990**	**Navsari**	**ba**	**9223423456**

Line and context addresses can also be mixed:

$ sed –n '1,/Joshi/p' student.lst

Using Regular Expressions

You can also use regular expression in context address.

```
$ sed –n ‘/Ch[ao][uw]*dha*ri/p’ student.lst
1112 | Ramesh S. Chaudhari | 20/02/1982 | Baroda  | bba  | 9926412126
1114 | Samir P. Chowdhari  | 12/02/1982 | Surat   | bcom | 8461626364
2234 | Jinal B. Chaudhri   | 17/07/1977 | Bharuch | bca  | 9427194271
3123 | Samira S. Chodhari  | 17/07/1977 | Bharuch | bcom | 9712323456
```

Display the mobile number start with 92.

```
$ sed –n ’/92••••••••$/p’ student.lst
2111 | Bimal R. Surati     | 10/08/1990 | Navsari | ba   | 9223423456
```

Writing selected lines to a file (**w**)

In sed, you can use the w (write) command to write the selected lines to a separate file.

save the lines of bca in separate file.

```
$ sed –n ‘/bca/w bca_list’ student.lst
```

We can also use multiple address like bba,bca,bcom,bba and store their lines in separate files.

```
$ sed –n  ‘/bca/w bca_list
        /bba/w bba_list
        /bcom/w bcom_list
        /ba/w ba_list’ student.lst
```

We can write the files using line addressing also.

```
$ sed –n ‘1,5w s1
           6,12w s2’ student.lst
```

Text Editing

Inserting lines

The **i** command inserts text. For example we can add the three line at the beginning of student.lst file.

```
$ sed ‘1i\
> This is Main data file in unix\
> it contains student information\
>it is ordinary file
> ’ student.lst > newstudent.lst
```

Deleting Lines (d)

sed uses the d (delete) command that select lines not containing the pattern. It is same as grep –v option.

```
$ sed '/bca/d' student.lst >other_list1
```

Or

```
$ sed –n '/bca/!p' student.lst > other_list2
```

Deleting blank lines

We can delete a blank line consist of any number of spaces, tabs or nothing.

```
$ sed '/^[  ]*$/d' blankfile
```

Press the [Tab] key or [Ctrl-i] inside the character class immediately after the space. Providing a ^ at the beginning and a $ at the end matches lines that contain only whitespace. This expression also matches lines containing nothing.

Substitutions (s)

s option of **sed** lets you replace a pattern with other pattern.

```
$ sed 's/|/:/' student.lst
```

The above command replace | with a : in the first occurance of every line. To replace all occurance of | with : in every line then use g (global) option.

```
$ sed 's/|/:/g' student.lst
```

To replace all occurance of | with : in selected line only use following:

```
$ sed '1,5s/|/:/g' student.lst
```

We can also replace word.

```
$ sed '1,5s/Surat/Daman/' student.lst
```

sed also uses regular expression for patterns to be substituted.

Replace Chaudhari, Chowdhari, Chaudhri, Chodhari with Chaudhari.

```
$ sed 's/Ch[ao][uw]*dh[a]*ri'/Chaudhari/' student.lst
```

sed also use the anchoring characters ^ and $.

If we want to add R at the beginning of roll no in student.lst then use the following

```
$ sed  's/^/R/' student.lst
```

If we want to add **(M)** at the end of mobile numer use the following.

```
$ sed 's/$/(M)/' student.lst
```

Performing Multiple Substitution

We can perform multiple substitutions with one sed command.

```
$ sed  's/11/51/

  > s/surat/baoda/

  > s/$/(M)' student.lst
```

Press enter after every substitution and end the quote after last substitution.

Compressing Multiple Spaces

We can remove multiple spaces which is at the end of second and third field in student.lst.

```
$ sed ‘s/ *|/|/g’ student.lst
```

The Remembered Pattern (//)

To store the scanned pattern (source pattern), the // is used. The // remembered the pattern.

```
$ sed ‘/ba/s//ptc/’ student.lst
```

The below three command do the same job.

```
$ sed ‘s/ba/ptc/’ student.lst

$ sed ‘/ba/s/ba/ptc/’ student.lst

        $ sed ‘/ba/s//ptc/’ student.lst
```

But the use of // in target pattern means you are removing the target pattern totally.

```
$ sed ‘s/|//g’ student.lst
```

We can Search the one pattern and can replace different pattern with other pattern.

```
$ sed –n ‘/bca/s/Bharuch/Bhavnagar/p’ student.lst
```

BASIC REGULAR EXPRESSION REVISITED

Both grep and sed uses regular expression.

Following are three types of expression:

1) The repeated Pattern
2) The interval regular expression (IRE)
3) The tagged regular expression (TRE)

The Repeated Pattern (&)

Sometime the source pattern also occurs in destination pattern. We can use special character **&** to represent it.

All the following three command do the same:

```
$ sed  ‘s/Surat/New Surat/’ student.lst

$ sed  ‘s/Surat/New &/’ student.lst

$ sed  ‘/Surat/s//New &’ student.lst
```

Interval Regular Expression (IRE)

IRE uses an integer to specify the number of characters preceding a pattern. The IRE uses an escaped pair of curly braces and takes three forms:

- ch\{m\} – The metacharacter ***ch*** can occur m times

- ch\{m,n\} – Here, ***ch*** can occur between m and n times.
- Ch\{m,\} – Here ***ch*** can occur atleast m times.

```
$ cat phonedir.txt

raj sharma 2526221

mohan patel 9898198921

raju soni 9321299332

mitali shah 2711222
```

Let’s display only those records which contains mobile number. We use IRE expression for it.

```
$ grep ‘[0-9]\{10\} phonedir.txt

mohan patel 9898198921

raju soni 9321299332
```

Extracting lines based on length

IRE is also used to select lines on the bases of length.

```
$ sed –n ‘/.\{10,\} f1              Display lines with atleast 10 characters

$ grep ‘^.\{10,20\}$’ f1            Display lines with length 10  to 20
```

The Tagged Regular Expression (TRE)

The TRE break a line into group and then extracts one or more of these groups. It requires two regular expression to be specified. One for source pattern and another is target pattern.

Display the lastname first than "," and first name and then mobile number.

```
$ sed 's/\([a-z]*\)  *\([a-z]*\)/\2,\1/' phonedir.txt

sharma, raj 2526221

patel, mohan 9898198921

soni, raju 9321299332

shah, mitali 2711222
```

display the record from student.lst with birth date in format yyyymmdd

```
$ sed 's^\(..\)/\(..\)/\(….\)^\3\2\1^' student.lst
```

We can choose different delimiter in sed command, but it must not occur in the entire command line. Here we use ^ to delimit patterns for substitution.

Table : Internal command used by sed

Command	**Description**
i, a, c	Inserts, appends and changes text
D	Delete line(s)

10q	Quits after reading the first 10 lines
p	Prints line(s) on standard output
3,$p	Prints lines 3 to end (-n option required)
$!p	Prints all lines except last line (-n option required)
/begin/,/end/p	Prints lines enclosed between begin and end (-n option required)
q	Quits after reading up to address line
r flname	Places contents of file flname after line
w flname	Write addressed lines to file flname
=	Prints line number addressed
s/s1/s2/	Replaces first occurrence of expression s1 in all lines with expression s2
10,20s/-/:/	Replaces first occurrence of **–** in lines 10 to 20 with a **:**
s/s1/s2/g	Replaces all occurrences of expression s1 in all lines with expression s2
s/-/:/g	Replaces all occurrences of **–** in all lines with a **:**