Como Usar o Comando Grep no Linux

Os sistemas operacionais Linux geralmente não possuem interfaces gráfica por motivos de segurança. Por isso, é muito importante saber como utilizar o terminal de comandos.

Uma das operações mais úteis que pode ser feita no terminal é a pesquisa dentro de um arquivo de texto, principalmente se você edita arquivos de configuração de serviço, como o NTP.

Neste tutorial, vamos ensinar como utilizar o **Comando Grep** no Linux (Unix) e mostrar exemplos práticos de sua funcionalidade.

Conteúdo

Como Usar o Comando Grep?

O Comando Grep é uma das ferramentas mais úteis e versáteis disponível no Unix. Ele procura por padrões especificados pelo usuário dentro de arquivos de texto. Em outras palavras, você pode pesquisar por palavras ou padrões e a linha ou linhas que serão exibidas.

Apesar de parecer um comando de pouca utilidade, sysadmins o utilizam com frequência para realizar pesquisas dentro dos diversos arquivos de configuração de um sistema.

Primeiro, conecte com o servidor VPS via SSH. Em caso de dúvidas, confira nosso tutorial.

Se você está em um computador Linux, basta abrir o terminal.

```
ssh usuario@seu-servidor
```

A sintaxe do Comando Grep ao realizar uma busca fica assim:

```
grep [opções] padrão [ARQUIVO]
```

- **grep** – a instrução de comando.
- **[opções]** – modificadores do comando.
- **padrão** – o que está sendo pesquisado.
- **[ARQUIVO]** – o arquivo em que a pesquisa está sendo realizada.

Você pode ver o manual e explicações de diversas opções do comando executando:

```
grep –help
```

O comando oferece muitas possibilidades, mas as mais comuns e mais utilizadas são:

- **-i** – a pesquisa não irá diferenciar letras maiúsculas de minúsculas. Ou seja, se você pesquisar por “carro” será o mesmo que pesquisar “CARRO”.
- **-c** – irá mostrar somente o número de linhas que combinam com o padrão pesquisado.
- **-r** – habilita pesquisa recursiva no diretório atual.
- **-v** – com esta opção, as linhas que não combinam com o padrão pesquisado serão exibidas.

Exemplos Úteis do Comando Grep Linux

Encontrando uma Palavra em um Arquivo de Texto

Para pesquisar uma palavra em um arquivo de texto basta executar o comando:

```
grep solicitação arquivo
```

- **solicitação** – a palavra que você está pesquisando.
- **arquivo** – o arquivo em que a busca está sendo realizada.

Em nosso caso, estamos procurando pela palavra **command** em um arquivo chamado **grep**:

```
grep command grep
```

O resultado destaca as linhas que combinam com a pesquisa, assim:

Encontre uma Palavra Ignorando Letras Maiúsculas ou Minúsculas

Para isso é necessário adicionar a opção **-i**.

```
grep -i solicitação arquivo
```

Simples assim!

Contador de Palavras

Com o comando grep você pode descobrir quantas vezes a palavra pesquisada aparece no arquivo de texto. Basta adicionar a opção **-c**.

```
grep -c solicitação arquivo
```

Pesquisando por Múltiplas Palavras

Até agora mostramos apenas exemplos em que pesquisamos uma única palavra. O grep também suporta pesquisas de várias palavras em um único comando, ficando assim:

```
grep solicitação1 arquivo | grep solicitação2 arquivo
```

O comando trabalha de maneira simples. Primeiro, pesquisamos pela **Solicitação1**, então o comando vai para a próxima palavra – **Solicitação2**.

Encontrando uma Palavra entre Vários Arquivos

Também é possível pesquisar em diversos arquivos por uma palavra em um único comando:

```
grep -l solicitação ./*
```

Os arquivos que contém a palavra que você pesquisou serão mostrados no resultado.

Conclusão

Usar o comando grep Linux torna a vida de quem trabalha com muitos arquivos de texto mais simples. Por isso, ele é considerado um dos comandos mais versáteis do sistema.

Neste tutorial, você aprendeu algumas das funções mais comuns do comando grep Linux. Também recomendamos um estudo do manual oficial para descobrir ainda mais funções e possibilidades!

https://www.hostinger.com.br/tutoriais/comando-grep-linux

The Grep Command Tutorial With Examples For Beginners

Written by sk *20601* Views
4 comments
5

In this tutorial, we are going to learn about **"grep"** command. Grep stands for **G**lobal **r**egular **e**xpression **p**rint. As the name implies, Grep is used to search text files with **regular expressions** (shortly **regex**). It prints the lines matching the given pattern in a text file. If no file is given, grep will recursively search the given pattern in the files in current directory. Grep has two variants, namely **egrep** and **fgrep**. These variants are deprecated but are provided for backward compatibility. Instead of using "grep -E" and "grep -F", you can use "egrp" and "fgrep", respectively. Without further ado, let us get started.

Grep Command examples

I have a file named **file.txt** with some random words. Let us have a look at file.txt:

```
$ cat file.txt
ostechnix
Ostechnix
o$technix
linux
linus
unix
technology
hello world
HELLO world
```

To begin the search, just type grep followed by what it is you're looking for and where you are looking from. For example, I am going to look for the string **"nix"** in file.txt. To do so, I run:

```
$ grep nix file.txt
```

Sample output:

```
ostechnix
Ostechnix
o$technix
unix
```

As you see in the above output, we got two words that contains the matching pattern "nix". If the search string has two words, mention them inside single quotes like below.

```
$ grep 'hello world' file.txt
hello world
```

You can also use **-n** flag to show the line numbers in the output:

```
$ grep -n nix file.txt
```

Sample output:

```
1:ostechnix
2:Ostechnix
3:o$technix
6:unix
```

This can be useful when you're working with a really long code.

Please note that **grep is case-sensitive**. For example, when you search for **"os"**, it is not going to display lines the contains uppercase letters i.e Os.

Check the following example.

```
$ grep os file.txt
ostechnix
```

But, I have another word named "Ostechnix" in file.txt, but grep didn't list it.

If you want, however, to ignore case sensitive, you can use **"-i"** flag like below.

```
$ grep -i os file.txt
ostechnix
Ostechn1x

$ grep -i 'hello world' file.txt
hello world
HELLO world
```

Now, grep didn't care about the case and we got the words that contains both uppercase and lowercase letters in the result.

We can also pipe an output of a command into grep. Have a look at the following example.

```
$ cat file.txt | grep os
ostechnix
```

Now see what we've got. The output of the file.txt is piped into grep and the words that contains the letters "os" in file.txt have been displayed.

We can also use some special characters or regular expressions in grep.

- **^** - search at the beginning of the line.
- **$** - search at the end of the line.
- **.** - Search any character.

Let me show you an example, so you can understand where and how to use those special characters.

You know already, we can search for the words that contains the string **"tech"** like below.

```
$ grep tech file.txt
ostechnix
Ostechnix
o$technix
technology
```

But what if you just wanted to search for the lines that only start with the word "tech". You don't want to display all the words that contains the string, but only the words that have the string "tech" at the beginning. It is also possible. This is where special characters comes in handy.

To search for the words that matches the pattern "tech" at the beginning of the line in a file, run:

```
$ grep ^tech file.txt
technology
```

Similarly, we can search for the words that ends with a particular letter(s), for example **"x"**, like below.

```
$ grep x$ file.txt
ostechnix
Ostechnix
o$technix
linux
unix
```

Also, you can find the words that contains any character using **.** (dot).

For example, let us search for any word that has **"n"** in the file.

```
$ grep .n file.txt
ostechnix
Ostechnix
o$technix
linux
linus
unix
technology
```

You should now have a basic understanding of grep usage. Let us go ahead and learn the other two variants, namely **egrep** and **fgrep**.

Egrep command examples

egrep stands for **e**xtended **grep**. It is similar to **"grep -E"** command. It will do all the things that grep will do. However, It provides some additional functionalities, such as using complicated regex, than the normal grep command does out of the box.

For instance, we can search for any words that start with either **"l"** or **"o"**.

Remember we use caret symbol (^) to search words at the beginning of line. Hence, our command for the above query will be:

```
$ egrep '^(l|o)' file.txt
ostechnix
o$technix
linux
linus
```

See? We have got all of the words that starts with either "l" or "o". Please note that the normal grep command can't do this.

Similarly, We can search for the lines that start with any character range between "l" to "u". That means, we will get the lines that start with l, m, n, o, p, q, r, s, t, and u. Everything else will be omitted from the result.

```
$ egrep '^[l-u]' file.txt
ostechnix
o$technix
linux
linus
unix
```

```
technology
```

Please note that I have used **square bracket** ([) to search for the range of words. Since grep is case-sensitive, it is not going to find lines that starts with uppercase letters in the given range.

To display all the lines that starts with both upper and lower case letters, we use:

```
$ egrep '^[l-u]|[L-U]' file.txt
```

Or,

```
$ egrep '^([l-u]|[L-U])' file.txt
```

Sample output:

```
ostechnix
Ostechnix
o$technix
linux
linus
unix
technology
HELLO world
```

See? Now we have got the words that begins with the character range "l" to "u" (either upper or lower case).

Fgrep command examples

fgrep stands for **f**ast **grep**. It is similar to **"grep -F"**. fgrep can't recognize regular expressions or special characters. fgrep can be used where you want regular expressions to be evaluated.

For example, we use the following command to find the words end in **"x"**.

```
$ grep x$ file.txt
ostechnix
Ostechnix
o$technix
linux
unix
```

Now run the same command with fgrep.

```
$ fgrep x$ file.txt
```

It will display nothing, because it couldn't evaluate the special characters.

Now let us see another example. To search for the words that matches the string **"o$"** with grep command, we do:

```
$ grep o$ file.txt
```

But, we get nothing, why? Because, as per the above command, we use the dollar symbol($) to find the words that ends in "o". Since there were no characters ends with "o" in file.txt, we get nothing.

Now, run the same command with fgrep.

```
$ fgrep o$ file.txt
o$technix
```

See? This is where we use fgrep command. It simply ignores the dollar symbol (the special character, of course) and displays the word that contains the string "o$".

For more details about grep, type:

```
$ grep --help
```

It will give all possible options. Or, refer the man pages.

```
$ man grep
```

Suggested read:

- **Learn To Use Man Pages Efficiently**

Etymology of grep

Grep was written overnight by **Ken Thompson**, the same guy who wrote Unix. But why and how did it get its name? Professor **Brian Kernighan** explains the etymology of grep in this video.

https://ostechnix.com/the-grep-command-tutorial-with-examples-for-beginners/

grep – Encontre trechos de código no terminal do Linux

Comando grep permite buscar palavras no código-fonte de vários arquivos simultaneamente no terminal do Linux; saiba como usar

Sobre o grep… Dia desses, enquanto passava a madrugada trabalhando, percebi que uma das tarefas era encontrar um trecho de código específico em mais de 2.000 arquivos e substituir por outro trecho de código. O problema é que não havia qualquer indicador óbvio de que o código em questão estaria em qualquer um desses arquivos.

- Canonical ensina como migrar do Windows 7 para o Ubuntu
- Por que o Google quer adotar o kernel Linux "puro" no Android?

Ou seja, eu provavelmente teria que baixar todos para minha máquina via FTP e abri-los para encontrar quais arquivos continham esse bendito trecho. Foi aí que eu me lembrei de um comando simples no Linux que pode ser executado no terminal e é uma mão na roda não só para usuários avançados, mas para qualquer um: o **grep!**

O que é grep?

O grep é um comando com uma função simples: ele procura por trechos de texto (strings) dentro de arquivos ou diretórios e retorna para você em quais arquivos a string foi encontrada, inclusive mostrando a linha em que isso ocorreu. Parece bobo? Pense novamente: as possibilidades com o grep são quase infinitas, mesmo para usuários comuns, e o uso aumenta muito mais quando você resolve uní-lo a outros comandos.

Imagine por exemplo que você queira descobrir rapidamente quais foram os últimos acessos de um número IP específico ao servidor. Em vez de abrir o arquivo de log ou usar uma ferramenta, você pode usar o grep no último arquivo de log de acessos. Pronto, ele te retorna na tela os dados.

Digamos que você não seja exatamente um usuário muito organizado e agora precisa encontrar onde está uma informação importante dentro de um documento perdido em meio a centenas de outros documentos. Use o grep. Ou, por fim, imagine que você seja um sysadmin que precisa urgentemente dormir pelo menos uma hora que seja e tem quase 2000 arquivos para verificar a existência de um possível trecho de código.

Use o grep, e vá dormir mais cedo.

Como o grep funciona?

No terminal, você pode digitar, por exemplo:

```
grep "trecho a procurar" arquivo.txt
```

Ou seja, primeiro o comando grep, logo depois o trecho que você quer encontrar, e por fim o arquivo onde você fará a busca. Embora não seja obrigatório colocar o trecho de busca entre aspas, o ideal é fazer isso sempre, para que o Linux não acabe se confundindo com a sintaxe do comando.

Mas… E se eu quiser procurar em vários arquivos, e não apenas em UM arquivos específico?

Não tema, continua simples:

```
grep "trecho a procurar" *
```

Com isso, o grep vai procurar o trecho em todos os arquivos do diretório atual. Se você quiser, pode especificar um diretório diferente:

```
grep "trecho a procurar" /var/www/*
```

Está ficando divertido, mas você ainda pode melhorar, com o uso de algumas opções de busca. Vamos dizer que você não queira fazer a busca apenas naquele diretório, mas também nos subdiretórios. Use a opção -R.

```
grep -R "trecho a procurar" /var/www/
```

Ele vai procurar o trecho em todos os arquivos e diretórios que estejam dentro de /var/www/. Perceba que, dependendo da quantidade de arquivos e do tamanho deles, isso pode levar alguns minutos ou várias horas. Então use essa opção com cuidado. Se você quiser, pode fazer uma busca que não se importe com maiúsculas ou minúsculas usando a opção -i:

```
grep -i "trecho a procurar" arquivo.txt
```

Aqui, o grep vai retornar qualquer resultado que bater com o trecho desejado, independente se houverem maiúsculas. Pode ser “Trecho A Procurar” ou “TrReChO a PrOcUrAr” ou qualquer outra combinação. Não fará diferença. Mas você também pode procurar por todos os arquivos ou linhas onde o trecho não aparece. Como? Usando a opção -v

```
grep -v "trecho a procurar" arquivo.txt
```

Nesse exemplo, o grep vai retornar qualquer linha do arquivo que não contenha o texto selecionado. Interessante, não? E dá para juntar todas essas opções? Claro que dá!

```
grep -Riv "trecho a procurar" /var/www/
```

Já ficou fácil de entender o que essa combinação faz, não? Procura recursivamente dentro da pasta /var/www por todos os arquivos e linhas onde o trecho selecionado não foi encontrado, independente dele ter partes em maiúsculas ou minúsculas.

Podemos melhorar? Podemos. E se o trecho que quisermos procurar existir em muitos arquivos e acabar “estourando” o limite de caracteres do terminal? Ou, melhor: e se quisermos passar o resultado para outra pessoa? É só jogar a saída do grep para um arquivo:

```
grep "trecho" arquivo.txt > arquivodesaida.txt
```

Todo o resultado do grep irá para o arquivodesaida.txt, e não para a tela. Assim, você poderá ler com calma, imprimir, enviar para alguém, ou até usar em algum outro script. Outra possibilidade é usar o grep para filtrar a saída de outros comandos. Vamos dizer que eu precise listar a saída do comando ls (muito útil se você está procurando algo específico em uma pasta com muitos arquivos):

```
ls | grep trecho
```

E pronto.

O ls vai retornar apenas os arquivos que contenham “trecho” no nome.

Mais fácil que isso, só se o estagiário fizer pra você!

O grep é uma ferramenta poderosa, e aqui cobrimos apenas o básico dela. Se você quiser saber mais, leia o manual do comando, e brinque um pouco na sua instalação Linux. Você vai descobrir possibilidades que tornarão sua vida muito mais fácil.

https://tecnoblog.net/70876/grep-tutorial-linux-codigo/

How To Use grep Command In Linux / UNIX With Practical Examples

Author: Vivek Gite Last updated: April 16, 2021 319 comments

How do I use grep command on Linux or Apple macOS/OS X? How can I use grep command on Unix operating systems? Can you give me a simple examples of the grep command?

Grep is an essential Linux and Unix command. It is used to search text and strings in a given file. In other words, grep command searches the given file for lines containing a match to the given strings or words. It is one of the most useful commands on Linux and Unix-like system for developers and sysadmins. Let us see how to use grep on a Linux or Unix like system.

Tutorial requirements

Requirements	Linux/Unix/macOS
Root privileges	No
Difficulty	Easy
	8 mintues

Table of contents

Did you know?

The name, “grep”, derives from the command used to perform a similar operation, using the Unix/Linux text editor ed:

```
g/re/p
```

The grep utilities are a family that includes grep, egrep, and fgrep for searching duties. For most uses, you need to use fgrep as it the fastest and only look into strings and words. However, typing grep is easy. Hence, it is a personal choice.

grep command examples in Linux and Unix

Below is some standard grep command explained with examples to get you started with grep on Linux, macOS, and Unix:

1. Search any line that contains the word in filename on Linux: **`grep 'word' filename`**
2. Perform a case-insensitive search for the word ‘bar’ in Linux and Unix: **`grep -i 'bar' file1`**
3. Look for all files in the current directory and in all of its subdirectories in Linux for the word ‘httpd’**`grep -R 'httpd' .`**
4. Search and display the total number of times that the string ‘nixcraft’ appears in a file named frontpage.md: **`grep -c 'nixcraft' frontpage.md`**

Let us see all commands and options in details.

Syntax

The syntax is as follows:

```
grep 'word' filename
fgrep 'word-to-search' file.txt
grep 'word' file1 file2 file3
grep 'string1 string2'  filename
cat otherfile | grep 'something'
command | grep 'something'
command option1 | grep 'data'
grep --color 'data' fileName
grep [-options] pattern filename
fgrep [-options] words file
```

How do I use grep to search a file on Linux?

Search /etc/passwd file for boo user, enter:

```
grep boo /etc/passwd
```

Sample outputs:

```
foo:x:1000:1000:boo,,,:/home/boo:/bin/ksh
```

We can use fgrep/grep to find all the lines of a file that contain a particular word. For example, to list all the lines of a file named address.txt in the current directory that contain the word “California”, run:

```
fgrep California address.txt
```

Please note that the above command also returns lines where “California” is part of other words, such as “Californication” or “Californian”. Hence pass the `-w` option with the grep/fgrep command to get only lines where “California” is included as a whole word:

```
fgrep -w California address.txt
```

You can force grep to ignore word case i.e match boo, Boo, BOO and all other combination with the `-i` option. For instance, type the following command:

```
grep -i "boo" /etc/passwd
```

The last `grep -i "boo" /etc/passwd` can run as follows using the cat command too:
```
cat /etc/passwd | grep -i "boo"
```

How to use grep recursively

You can search recursively i.e. read all files under each directory for a string “192.168.1.5”
```
$ grep -r "192.168.1.5" /etc/
```
OR
```
$ grep -R "192.168.1.5" /etc/
```
Sample outputs:

```
/etc/ppp/options:# ms-wins 192.168.1.50
/etc/ppp/options:# ms-wins 192.168.1.51
/etc/NetworkManager/system-connections/Wired connection
1:addresses1=192.168.1.5;24;192.168.1.2;
```

You will see result for 192.168.1.5 on a separate line preceded by the name of the file (such as /etc/ppp/options) in which it was found. The inclusion of the file names in the output data can be suppressed by using the `-h option` as follows:
```
$ grep -h -R "192.168.1.5" /etc/
```
OR
```
$ grep -hR "192.168.1.5" /etc/
```
Sample outputs:

```
# ms-wins 192.168.1.50
# ms-wins 192.168.1.51
addresses1=192.168.1.5;24;192.168.1.2;
```

How to use grep to search words only

When you search for boo, grep will match fooboo, boo123, barfoo35 and more. You can force the grep command to select only those lines containing matches that form whole words i.e. match only boo word:
```
$ grep -w "boo" file
```

How to use grep to search 2 different words

Use the egrep command as follows:
```
$ egrep -w 'word1|word2' /path/to/file
```

Ignore case

We can force grep to ignore case distinctions in patterns and data. For example, when I search for ‘bar’, match ‘BAR’, ‘Bar’, ‘BaR’ and so on:
```
$ grep -i 'bar' /path/to/file
```
In this example, I am going to include all subdirectories in a search:
```
$ grep -r -i 'main' ~/projects/
```

How can I count line when words has been matched

The grep can report the number of times that the pattern has been matched for each file using `-c` (count) option:

```
$ grep -c 'word' /path/to/file
```

Pass the `-n` option to precede each line of output with the number of the line in the text file from which it was obtained:

```
$ grep -n 'root' /etc/passwd
```

```
1:root:x:0:0:root:/root:/bin/bash
1042:rootdoor:x:0:0:rootdoor:/home/rootdoor:/bin/csh
3319:initrootapp:x:0:0:initrootapp:/home/initroot:/bin/ksh
```

Force grep invert match

You can use `-v` option to print inverts the match; that is, it matches only those lines that do not contain the given word. For example print all line that do not contain the word bar:

```
$ grep -v bar /path/to/file
$ grep -v '^root' /etc/passwd
```

Display lines before and after the match

Want to see the lines before your matches? Try passing the `-B` to the grep:

```
grep -B NUM "word" file
grep -B 3 "foo" file1
```

Similarly, display the lines after your matches by passing the `-A` to the grep:

```
grep -A NUM "string" /pth/to/file
grep -A 4 "dropped" /var/log/ufw.log
```

We can combine those two options to get most meaningful outputs:

```
grep -C 4 -B 5 -A 6 --color 'error-code' /var/log/httpd/access_log
```

Here is a sample shell script that fetches the Linux kernel download urls:

```
.......
...
_out="/tmp/out.$$"
curl -s https://www.kernel.org/ > "$_out"
########################
## grep -A used here ##
########################
url="$(grep -A 2 '<td id="latest_button">' ${_out}  | grep -Eo
'(http|https)://[^/"]+.*xz')"
gpgurl="${url/tar.xz/tar.sign}"
notify-send "A new kernel version ($remote) has been released."
echo "* Downloading the Linux kernel (new version) ..."
wget -qc "$url" -O "${dldir}/${file}"
wget -qc "$gpgurl" -O "${dldir}/${gpgurl##*/}"
.....
..
```

UNIX / Linux pipes

grep command often used with shell pipes. In this example, show the name of the hard disk devices:

```
# dmesg | egrep '(s|h)d[a-z]'
```

Display cpu model name:

```
# cat /proc/cpuinfo | grep -i 'Model'
```

However, above command can be also used as follows without shell pipe:

```
# grep -i 'Model' /proc/cpuinfo
```

```
model           : 30
model name      : Intel(R) Core(TM) i7 CPU       Q 820  @ 1.73GHz
model           : 30
model name      : Intel(R) Core(TM) i7 CPU       Q 820  @ 1.73GHz
```

One of my favorite usage of grep or egrep command to filter the output of the yum command/dpkg command/apt command/apt-get command:

```
dpkg --list | grep linux-image
yum search php | grep gd
apt search maria | egrep 'server|client'
```

Linux grep commands explained with shell pipes examples

How do I list just the names of matching files?

Use the `-l` option to list file name whose contents mention main():

```
$ grep -l 'main' *.c
```

OR

```
$ grep -Rl 'main' /path/to/project/dir/
```

Colors option

Finally, we can force grep to display output in colors, enter:

```
$ grep --color vivek /etc/passwd
```

Grep command in action

In conclusion, the `--color` option increase readability. For example, the `GREP_COLOR` environment variable and the `grep --color=always` can be used as follows:

```
GREP_COLOR='1;35' grep --color=always 'vivek' /etc/passwd
GREP_COLOR='1;32' grep --color=always 'vivek' /etc/passwd
GREP_COLOR='1;37' grep --color=always 'root' /etc/passwd
GREP_COLOR='1;36' grep --color=always nobody /etc/passwd
```

In addition, to default red color now we can define colors using GREP_COLOR shell variable. The differnt color helps us massivly with visual grepping.

Conclusion

The grep command is a very versatile and many new Linux or Unix users find it complicated. Hence, I suggest you read the grep man page too. Let us summarize most import options:

grep command in Unix/Linux

Options	Description
`-i`	Ignore case distinctions on Linux and Unix
`-w`	Force PATTERN to match only whole words
`-v`	Select non-matching lines
`-n`	Print line number with output lines
`-h`	Suppress the Unix file name prefix on output
`-r`	Search directories recursivly on Linux
`-R`	Just like -r but follow all symlinks
`-l`	Print only names of FILEs with selected lines
`-c`	Print only a count of selected lines per FILE
`--color`	Display matched pattern in colors

If you enjoyed the grep tutorial, then you might like to read our “Regular Expressions in Grep” tutorial. Read grep command man page by typing the following man command:

```
man grep
```

https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/