

grep

????

- <http://www.cyberciti.biz/faq/grep-regular-expressions/>
- <https://www.ubuntupit.com/practical-grep-command-for-linux-enthusiasts/>

????

```
# . (period)
> grep dev.sda /etc/fstab
/dev/sda3    /          reiserfs    noatime,ro 1 1
/dev/sda1    /boot      reiserfs    noauto,noatime,notail 1 2
/dev/sda2    swap       swap        sw 0 0
#/dev/sda4   /mnt/extra reiserfs     noatime,rw 1 1

# [ ]
> grep dev.sda[12] /etc/fstab
/dev/sda1    /boot      reiserfs    noauto,noatime,notail 1 2
/dev/sda2    swap       swap        sw 0 0

# [^12] [1,2]
> grep dev.sda[^12] /etc/fstab
/dev/sda3    /          reiserfs    noatime,ro 1 1
#/dev/sda4   /mnt/extra reiserfs     noatime,rw 1 1

# [ ]
> grep '^#' /etc/fstab

# /etc/fstab: static file system information.
#

> grep '^#.*$' /etc/fstab
# /etc/fstab: static file system information.
#
```

```
> grep 'foo$' filename
```

#

```
> echo "AAA BBB ccc ddd" | grep -wE -o "(cc|BBB|ddd)"
```

[illegible]

```
> grep -w "boo" myfile.txt
```

```
> grep "\<boo\>" myfile.txt
```


--	--	--	--	--	--	--

--	--	--

```
> grep '\*\*\*' myfile.txt
```

email

```
> grep -E -o "\b[A-Za-z0-9._%+~]@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}\b" /path/to/data
```

, 1-XXX-ZZZ-YYYY, XXX-ZZZ-YYYY

```
> grep -E '(1)?[[:digit:]]{3}-[[:digit:]]{3}-[[:digit:]]{4}' sample.txt
```

IP

```
> egrep '\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)' /etc/hosts
```

```
> grep -E -o "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)" /var/log/auth.log
```


--	--	--	--	--	--

```
> grep '[vV][il][Vv][Ee][kK]' filename
```

```
> grep '[vV]ivek' filename
```

```
> grep -w '[vV]ivek[0-9]' filename
```

```
> grep 'foo[0-9][0-9]' filename
```

```
> grep '[A-Za-z]' filename
```

```
> grep [wn] filename
```

?????

> ???????(???????)

```
ls /var/log/ | grep -E "http|secure"
```

```
ls /var/log/ | grep "http\|secure"
```

```
grep -Ev "^(#|;)" example.txt
```

> ?????????????

???????????

```
cd /var/www/html
grep -r "[\[\]\{\}\*\]" *

grep -Ril "specific_text" /path/to/dir
```

> ?????????????

????????????????? AIX ? `grep -p`

???: [grepp.awk](#)

> ?????????????

```
# Before n lines
grep -B1 "^SQL" runstats.log

# After n lines
grep -A1 "^SQL" runstats.log

# Without separator '--'
grep -B1 --no-group-separator "^SQL" runstats.log
```

?????????

```
CONFIG_CHECK=`grep "^# SparkleShare$" /etc/ssh/sshd_config`
if ! [ "$CONFIG_CHECK" = "# SparkleShare" ]; then
    echo "" >> /etc/ssh/sshd_config
    echo "# SparkleShare" >> /etc/ssh/sshd_config
    echo "Match User storage" >> /etc/ssh/sshd_config
    echo "    PasswordAuthentication no" >> /etc/ssh/sshd_config
fi
```

?? grep ?????? ps ?

```
↪ ps -ef | grep 'plank'                                2022-04-24 (三) 09:30:51 CST
alang      2575  2166  0 09:22 ?        00:00:02 plank
```

```
alang 4831 3916 0 09:30 pts/0 00:00:00 grep --color=auto plank <=====
```

```
→ ps -ef | grep '[p]lank' [2022-04-24 (09:30:55) CST
```

```
alang 2575 2166 0 09:22 ? 00:00:02 plank
```

????

```
# [ ] [ ] [ ] [ ] [ ] [ ]
```

```
while read line; do grep -w "^${line%%.*}" /etc/shadow; done <passwd.bak >shadow.bak
```

Cheat Sheet

GREP CHEATSHEET

Usage

```
grep [OPTION...] PATTERN [FILE...]
grep [OPTION...] -e PATTERN ... [FILE...]
grep [OPTION...] -F PATTERN_FILE ... [FILE...]
```

Description

grep searches each FILE for PATTERN. PATTERN is a string that contains one or more patterns separated by newline characters, and grep prints each line that matches a pattern. When using grep in a shell command, PATTERN should usually be quoted.

A FILE of "-" denotes standard input. If no FILE is specified, recursive searches examine the working directory, while nonrecursive searches read standard input.

Furthermore, the variant programs egrep, fgrep, and rgrep are equivalent to grep -E, grep -F, and grep -r, respectively. These variants are obsolete, but remain available for backward compatibility.

Options Examples

```
grep 'alias' filename          find all occurrences of a string (basic usage)
-c grep -c 'error' /var/log/syslog count the number of matches
-e grep -e 'linux*' filename    use regex (lines ending with 'Linux')
-f grep -f pattern_file .bashrc take PATTERN from FILE
-i grep -i 'Alias' .bashrc      ignore case search
-l grep -l 'alias' /home/trau/* returns files with matches
-L grep -L 'alias' /home/trau/* returns files without matches
-r grep -r 'error' /var/log/log.txt recursive search (within subdirs)
-v grep -v 'warning' /home/trau/log.txt select non-matching lines
-M grep -m 3 'alias' .bashrc    limit grep output upto 3 lines.
-n grep -n 'sudo' .bashrc       print line numbers of the matches.
-o grep -o 'search string' .bashrc only show the matching part of the string.
-x grep -x 'linux is love' filename match only whole lines
-w grep -w 'alias' .bashrc      match only whole words
-A grep -A 6 'error' error.log  print 6 lines after a match
-B grep -B 2 'warning' error.log print 2 lines before a match
-C grep -C 7 'info' error.log   print 7 lines before and after a match
-E grep -E 'b(a|e)g' filename  Extended regex (lines containing bag or bag)
```

Grep Regular Expressions

Wildcards

.	Match any Character Except New Line
?	Match 0 or One characters
+	Match 0 or More characters
*	Match 1 or More characters

Characters

[A-Za-z]	Any lower and upper case letter.
[0-9]	Any number.
[0-9A-Za-z]	Any lower and upper case letter or digit.

BRE, ERE & PCRE

Basic Regular Expressions (BRE)

Unless they are preceded by a backslash, these characters have a special meaning in BRE: . * [] \

However, unless they are preceded by a backslash, these characters have no special meaning: ? + () | { }

Extended Regular Expressions (ERE)

Unless they are escaped with a backslash, ERE gives each of these characters a unique meaning: . * + ? [] () | { }

Perl Compatible Regular Expressions (PCRE)

Additional anchors and character classes, lookahead/lookbehind, conditional expressions, comments, and other features are available in PCRE.

Quantifiers

{n}	Matches exactly n times
{n,}	Matches n or More
{,n}	matches up to n (maximum)
{n,m}	Matches between n and m (Minimum, Maximum)

Posix

```
[:alpha:] any alphabetical character, either upper or lower case
[:alnum:] Matches any alphanumeric character 0-9, A-Z, or a-z
[:blank:] Matches a space or Tab character
[:digit:] Matches a numerical digit from 0 through 9
[:lower:] Matches any lowercase alphabetical character a-z
[:print:] Matches any printable character
[:punct:] Matches a punctuation character
[:upper:] Matches any uppercase alphabetical character A-Z
[:space:] Matches any whitespace character: space, Tab, NL (newline), FF (formfeed), VT (vertical tab), CR (carriage return)
```

Position

^	Beginning of line.
\$	End of line.
^\$	Empty line.
\b	Start of word.
\b	End of word.



Revision #15

Created 1 June 2020 06:15:30 by Admin

Updated 18 October 2024 15:52:50 by Admin