**Using Grep & Regular Expressions to Search for Text Patterns in Linux**

Jul 22, 2013 [Linux Basics](https://www.digitalocean.com/community/tags/linux-basics?type=tutorials)

**Introduction**

One of the most useful and versatile commands in a Linux terminal environment is the "grep" command. The name "grep" stands for "global regular expression print". This means that grep can be used to see if the input it receives matches a specified pattern.

This seemingly trivial program is extremely powerful when used correctly. Its ability to sort input based on complex rules makes it a popular link in many command chains.

We will explore some options and then dive into using regular expressions. All of the techniques discussed in this guide can be applied to managing your VPS server.

**Table of Contents**

1. [Basic Usage](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#basic-usage)
   * [Common Options](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#common-options)
2. [Regular Expressions](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#regular-expressions)
   * [Literal Matches](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#literal)
   * [Anchor Matches](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#anchor)
   * [Matching Any Character](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#any-character)
   * [Bracket Expressions](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#bracket)
   * [Repeat Pattern Zero or More Times](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#repeat-zero-or-more)
   * [Escaping Meta-Characters](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#escaping-meta)
3. [Extended Regular Expressions](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#extended-regex)
   * [Grouping](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#grouping)
   * [Alternation](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#alternation)
   * [Quantifiers](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#quantifiers)
   * [Specifying Match Repetitions](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#match-repetitions)
4. [Conclusion](https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux#conclusion)

**Basic Usage**

In its simplest form, grep can be used to match literal patterns within a text file. This means that if you pass grep a word to search for, it will print out every line in the file containing that word.

Let's try an example. We will use grep to search for every line that contains the word "GNU" in the GNU General Public License version 3 on an Ubuntu system.

cd /usr/share/common-licenses

grep "GNU" GPL-3

GNU GENERAL PUBLIC LICENSE

The GNU General Public License is a free, copyleft license for

the GNU General Public License is intended to guarantee your freedom to

GNU General Public License for most of our software; it applies also to

Developers that use the GNU GPL protect your rights with two steps:

"This License" refers to version 3 of the GNU General Public License.

13. Use with the GNU Affero General Public License.

under version 3 of the GNU Affero General Public License into a single

...

...

The first argument, "GNU", is the pattern we are searching for, while the second argument, "GPL-3", is the input file we wish to search.

The resulting output will be every line containing the pattern text. In some Linux distributions, the searched for pattern will be highlighted in the resulting lines.

**Common Options**

By default, grep will simply search for the exact specified pattern within the input file and return the lines it finds. We can make this behavior more useful though by adding some optional flags to grep.

If we would want grep to ignore the "case" of our search parameter and search for both upper- and lower-case variations, we can specify the "-i" or "--ignore-case" option.

We will search for each instance of the word "license" (with upper, lower, or mixed cases) in the same file as before.

grep -i "license" GPL-3

GNU GENERAL PUBLIC LICENSE

of this license document, but changing it is not allowed.

The GNU General Public License is a free, copyleft license for

The licenses for most software and other practical works are designed

the GNU General Public License is intended to guarantee your freedom to

GNU General Public License for most of our software; it applies also to

price. Our General Public Licenses are designed to make sure that you

(1) assert copyright on the software, and (2) offer you this License

"This License" refers to version 3 of the GNU General Public License.

"The Program" refers to any copyrightable work licensed under this

...

...

As you can see, we have been given results that contain: "LICENSE", "license", and "License". If there was an instance with "LiCeNsE", that would have been returned as well.

If we want to find all lines that **do not** contain a specified pattern, we can use the "-v" or "--invert-match" option.

We can search for every line that does not contain the word "the" in the BSD license with the following command:

grep -v "the" BSD

All rights reserved.

Redistribution and use in source and binary forms, with or without

are met:

may be used to endorse or promote products derived from this software

without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND

ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

...

...

As you can see, because we did not specify the "ignore case" option, the last two items were returned as not having the word "the".

It is often useful to know the line number that the matches occur on. This can be accomplished by using the "-n" or "--line-number" option.

The previous example with this flag added would return the following text:

grep -vn "the" BSD

2:All rights reserved.

3:

4:Redistribution and use in source and binary forms, with or without

6:are met:

13: may be used to endorse or promote products derived from this software

14: without specific prior written permission.

15:

16:THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND

17:ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

...

...

Now we can reference the line number if we want to make changes to every line that does not contain "the".

**Regular Expressions**

In the introduction, we stated that grep stands for "global regular expression print". A "regular expression" is a text string that describes a particular search pattern.

Different applications and programming languages implement regular expressions slightly differently. We will only be exploring a small subset of the way that grep describes its patterns.

**Literal Matches**

The examples above, when we searched for the words "GNU" and "the", we were actually searching for very simple regular expressions, which matched the exact string of characters "GNU" and "the".

It is helpful to always think of these as matching a string of characters rather than matching a word. This will become a more important distinction as we learn more complex patterns.

Patterns that exactly specify the characters to be matched are called "literals" because they match the pattern literally, character-for-character.

All alphabetic and numerical characters (as well as certain other characters) are matched literally unless modified by other expression mechanisms.

**Anchor Matches**

Anchors are special characters that specify where in the line a match must occur to be valid.

For instance, using anchors, we can specify that we only want to know about the lines that match "GNU" at the very beginning of the line. To do this, we could use the "^" anchor before the literal string.

This string example will only mach "GNU" if it occurs at the very beginning of a line.

grep "^GNU" GPL-3

GNU General Public License for most of our software; it applies also to

GNU General Public License, you may choose any version ever published

Similarly, the "$" anchor can be used after a string to indicate that the match will only be valid if it occurs at the very end of a line.

We will match every line ending with the word "and" in the following regular expression:

grep "and$" GPL-3

that there is no warranty for this free software. For both users' and

The precise terms and conditions for copying, distribution and

License. Each licensee is addressed as "you". "Licensees" and

receive it, in any medium, provided that you conspicuously and

alternative is allowed only occasionally and noncommercially, and

network may be denied when the modification itself materially and

adversely affects the operation of the network or violates the rules and

provisionally, unless and until the copyright holder explicitly and

receives a license from the original licensors, to run, modify and

make, use, sell, offer for sale, import and otherwise run, modify and

**Matching Any Character**

The period character (.) is used in regular expressions to mean that any single character can exist at the specified location.

For example, if we want to match anything that has two characters and then the string "cept", we could use the following pattern:

grep "..cept" GPL-3

use, which is precisely where it is most unacceptable. Therefore, we

infringement under applicable copyright law, except executing it on a

tells the user that there is no warranty for the work (except to the

License by making exceptions from one or more of its conditions.

form of a separately written license, or stated as exceptions;

You may not propagate or modify a covered work except as expressly

9. Acceptance Not Required for Having Copies.

...

...

As you can see, we have instances of both "accept" and "except" and variations of the two words. The pattern would also have matched "z2cept" if that was found as well.

**Bracket Expressions**

By placing a group of characters within brackets ("[" and "]"), we can specify that the character at that position can be any one character found within the bracket group.

This means that if we wanted to find the lines that contain "too" or "two", we could specify those variations succinctly by using the following pattern:

grep "t[wo]o" GPL-3

your programs, too.

freedoms that you received. You must make sure that they, too, receive

Developers that use the GNU GPL protect your rights with two steps:

a computer network, with no transfer of a copy, is not conveying.

System Libraries, or general-purpose tools or generally available free

Corresponding Source from a network server at no charge.

...

...

We can see that both variations are found within the file.

Bracket notation also allows us some interesting options. We can have the pattern match anything **except** the characters within a bracket by beginning the list of characters within the brackets with a "^" character.

^ within the brackets means except. For example: grep “[^c]ode” GPL-3

^ outside the brackets means the very beginning. For example: grep “^code” GPL-3

This example is like the pattern ".ode", but will not match the pattern "code":

grep "[^c]ode" GPL-3

1. Source Code.

model, to give anyone who possesses the object code either (1) a

the only significant mode of use of the product.

notice like this when it starts in an interactive mode:

You will notice that in the second line returned, there is, in fact, the word "code". This is not a failure of the regular expression or grep.

Rather, this line was returned because earlier in the line, the pattern "mode", found within the word "model", was found. The line was returned because there was an instance that matched the pattern.

Another helpful feature of brackets is that you can specify a range of characters instead of individually typing every available character.

This means that if we want to find every line that begins with a capital letter, we can use the following pattern:

grep "^[A-Z]" GPL-3

GNU General Public License for most of our software; it applies also to

States should not allow patents to restrict development and use of

License. Each licensee is addressed as "you". "Licensees" and

Component, and (b) serves only to enable use of the work with that

Major Component, or to implement a Standard Interface for which an

System Libraries, or general-purpose tools or generally available free

Source.

User Product is transferred to the recipient in perpetuity or for a

...

...

Due to some legacy sorting issues, it is often more accurate to use POSIX character classes instead of character ranges like we just used.

There are many character classes that are outside of the scope of this guide, but an example that would accomplish the same procedure as above uses the "[:upper:]" character class within a bracket selector:

grep "^[[:upper:]]" GPL-3

GNU General Public License for most of our software; it applies also to

States should not allow patents to restrict development and use of

License. Each licensee is addressed as "you". "Licensees" and

Component, and (b) serves only to enable use of the work with that

Major Component, or to implement a Standard Interface for which an

System Libraries, or general-purpose tools or generally available free

Source.

User Product is transferred to the recipient in perpetuity or for a

...

...

**Repeat Pattern Zero or More Times**

Finally, one of the most commonly used meta-characters is the "\*", which means "repeat the previous character or expression zero or more times".

If we wanted to find each line that contained an opening and closing parenthesis, with only letters and single spaces in between, we could use the following expression:

grep "([A-Za-z ]\*)" GPL-3

Copyright (C) 2007 Free Software Foundation, Inc.

distribution (with or without modification), making available to the

than the work as a whole, that (a) is included in the normal form of

Component, and (b) serves only to enable use of the work with that

(if any) on which the executable work runs, or a compiler used to

(including a physical distribution medium), accompanied by the

(including a physical distribution medium), accompanied by a

place (gratis or for a charge), and offer equivalent access to the

...

...

**Escaping Meta-Characters**

Sometimes, we may want to search for a literal period or a literal opening bracket. Because these characters have special meaning in regular expressions, we need to "escape" these characters to tell grep that we do not wish to use their special meaning in this case.

We can escape characters by using the backslash character (\) before the character that would normally have a special meaning.

For instance, if we want to find any line that begins with a capital letter and ends with a period, we could use the following expression. The ending period is escaped so that it represents a literal period instead of the usual "any character" meaning:

grep "^[A-Z].\*\.$" GPL-3

Source.

License by making exceptions from one or more of its conditions.

License would be to refrain entirely from conveying the Program.

ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

SUCH DAMAGES.

Also add information on how to contact you by electronic and paper mail.

**Extended Regular Expressions**

Grep can be used with an even more extensive regular expression language by using the "-E" flag or by calling the "egrep" command instead of grep.

These options open up the capabilities of "extended regular expressions". Extended regular expressions include all of the basic meta-characters, along with additional meta-characters to express more complex matches.

**Grouping**

One of the easiest and most useful abilities that extended regular expressions open up is the ability to group expressions together to manipulate or reference as one unit.

Group expressions together using parentheses. If you would like to use parentheses without using extended regular expressions, you can escape them with the backslash to enable this functionality.

grep "\(grouping\)" file.txt

grep -E "(grouping)" file.txt

egrep "(grouping)" file.txt

The above three expressions are functionally equivalent.

**Alternation**

Similar to how bracket expressions can specify different possible choices for single character matches, alternation allows you to specify alternative matches for strings or expression sets.

To indicate alternation, we use the pipe character "|". These are often used within parenthetical grouping to specify that one of two or more possibilities should be considered a match.

The following will find either "GPL" or "General Public License" in the text:

grep -E "(GPL|General Public License)" GPL-3

The GNU General Public License is a free, copyleft license for

the GNU General Public License is intended to guarantee your freedom to

GNU General Public License for most of our software; it applies also to

price. Our General Public Licenses are designed to make sure that you

Developers that use the GNU GPL protect your rights with two steps:

For the developers' and authors' protection, the GPL clearly explains

authors' sake, the GPL requires that modified versions be marked as

have designed this version of the GPL to prohibit the practice for those

...

...

Alternation can select between more than two choices by adding additional choices within the selection group separated by additional pipe (|) characters.

**Quantifiers**

Like the "\*" meta-character, that matched the previous character or character set zero or more times, there are other meta-characters available in extended regular expressions that specify the number of occurrences.

To match a character zero or one times, you can use the "?" character. This makes character or character set that came before optional, in essence.

The following matches "copyright" and "right" by putting "copy" in an optional group:

grep -E "(copy)?right" GPL-3

Copyright (C) 2007 Free Software Foundation, Inc.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have

know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License

"Copyright" also means copyright-like laws that apply to other kinds of

...

...

The "+" character matches an expression one or more times. This is almost like the "\*" meta-character, but with the "+" character, the expression *must* match at least once.

The following expression matches the string "free" plus one or more characters that are not whitespace:

grep -E "free[^[:space:]]+" GPL-3

The GNU General Public License is a free, copyleft license for

to take away your freedom to share and change the works. By contrast,

the GNU General Public License is intended to guarantee your freedom to

When we speak of free software, we are referring to freedom, not

have the freedom to distribute copies of free software (and charge for

you modify it: responsibilities to respect the freedom of others.

freedoms that you received. You must make sure that they, too, receive

protecting users' freedom to change the software. The systematic

of the GPL, as needed to protect the freedom of users.

patents cannot be used to render the program non-free.

**Specifying Match Repetition**

If we need to specify the number of times that a match is repeated, we can use the brace characters ("{" and "}"). These characters are used to specify an exact number, a range, or an upper or lower bounds to the amount of times an expression can match.

If we want to find all of the lines that contain triple-vowels, we can use the following expression:

grep -E "[AEIOUaeiou]{3}" GPL-3

changed, so that their problems will not be attributed erroneously to

authors of previous versions.

receive it, in any medium, provided that you conspicuously and

give under the previous paragraph, plus a right to possession of the

covered work so as to satisfy simultaneously your obligations under this

If we want to match any words that have between 16 and 20 characters, we can use the following expression:

grep -E "[[:alpha:]]{16,20}" GPL-3

certain responsibilities if you distribute copies of the software, or if

you modify it: responsibilities to respect the freedom of others.

c) Prohibiting misrepresentation of the origin of that material, or

**Conclusion**

There are many times when grep will be useful in finding patterns within files or within the file system hierarchy. It is worthwhile to become familiar with its options and syntax to save yourself time when you need it.

Regular expressions are even more versatile, and can be used with many popular programs. For instance, many text editors implement regular expressions for searching and replacing text.

Furthermore, most modern programming languages use regular expressions to perform procedures on specific pieces of data. Regular expressions are a skill that will be transferrable to many common computer-related tasks.