grep

强大的文本搜索工具

补充说明

grep （global search regular expression(RE) and print out the line，全面搜索正则表达式并把行打印出来）是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。用于过滤/搜索的特定字符。可使用正则表达式能配合多种命令使用，使用上十分灵活。

选项

```
-a --text  # 不要忽略二进制数据。
-A <显示行数>   --after-context=<显示行数> # 除了显示符合范本样式的那一行之外，并显示该行之后的内容。
-b --byte-offset                                # 在显示符合范本样式的那一行之外，并显示该行之前的内容。
-B<显示行数>   --before-context=<显示行数> # 除了显示符合样式的那一行之外，并显示该行之前的内容。
-c --count    # 计算符合范本样式的列数。
-C<显示行数> --context=<显示行数>或-<显示行数> # 除了显示符合范本样式的那一列之外，并显示该列之前后的内容。
-d<进行动作> --directories=<动作> # 当指定要查找的是目录而非文件时，必须使用这项参数，否则grep命令将回报信息并停止动作。
-e<范本样式> --regexp=<范本样式> # 指定字符串作为查找文件内容的范本样式。
-E --extended-regexp             # 将范本样式为延伸的普通表示法来使用，意味着使用能使用扩展正则表达式。
-f<范本文件> --file=<规则文件>  # 指定范本文件，其内容有一个或多个范本样式，让grep查找符合范本条件的文件内容，格式为每一列的范本样式。
-F --fixed-regexp   # 将范本样式视为固定字符串的列表。
-G --basic-regexp   # 将范本样式视为普通的表示法来使用。
-h --no-filename    # 在显示符合范本样式的那一列之前，不标示该列所属的文件名称。
-H --with-filename  # 在显示符合范本样式的那一列之前，标示该列的文件名称。
-i --ignore-case    # 忽略字符大小写的差别。
-l --file-with-matches   # 列出文件内容符合指定的范本样式的文件名称。
-L --files-without-match # 列出文件内容不符合指定的范本样式的文件名称。
-n --line-number         # 在显示符合范本样式的那一列之前，标示出该列的编号。
-P --perl-regexp         # PATTERN 是一个 Perl 正则表达式
-q --quiet或--silent     # 不显示任何信息。
-R/-r  --recursive       # 此参数的效果和指定“-d recurse”参数相同。
-s --no-messages  # 不显示错误信息。
-v --revert-match # 反转查找。
-V --version      # 显示版本信息。
-w --word-regexp  # 只显示全字符合的列。
-x --line-regexp  # 只显示全列符合的列。
-y # 此参数效果跟“-i”相同。
-o # 只输出文件中匹配到的部分。
-m <num> --max-count=<num> # 找到num行结果后停止查找，用来限制匹配行数
```

规则表达式

```
^  # 锚定行的开始 如：'^grep'匹配所有以grep开头的行。
```

```
$  # 锚定行的结束 如□'grep$' 匹配所有以grep结尾的行。
.  # 匹配一个非换行符的字符 如□'gr.p'匹配gr后接一个任意字符，然后是p□
*  # 匹配零个或多个先前字符 如□'*grep'匹配所有一个或多个空格后紧跟grep的行。
.* # 一起用代表任意字符。
[] # 匹配一个指定范围内的字符，如'[Gg]rep'匹配Grep和grep□
[^] # 匹配一个不在指定范围内的字符，如□'[^A-FH-Z]rep'匹配不包含A-R和T-Z的一个字母开头，紧跟rep的行。
\(..\) # 标记匹配字符，如'\(love\)'□love被标记为1。
\<   # 锚定单词的开始，如:'\<grep'匹配包含以grep开头的单词的行。
\>   # 锚定单词的结束，如'grep\>'匹配包含以grep结尾的单词的行。
x\{m\}  # 重复字符x□m次，如：'0\{5\}'匹配包含5个o的行。
x\{m,\}   # 重复字符x,至少m次，如□'o\{5,\}'匹配至少有5个o的行。
x\{m,n\}  # 重复字符x□至少m次，不多于n次，如□'o\{5,10\}'匹配5--10个o的行。
\w    # 匹配文字和数字字符，也就是[A-Za-z0-9]□如：'G\w*p'匹配以G后跟零个或多个文字或数字字符，然后是p□
\W    # \w的反置形式，匹配一个或多个非单词字符，如点号句号等。
\b    # 单词锁定符，如: '\bgrep\b'只匹配grep□
```

grep命令常见用法

在文件中搜索一个单词，命令会返回一个包含 **“match_pattern”** 的文本行：

```
grep match_pattern file_name
grep "match_pattern" file_name
```

在多个文件中查找：

```
grep "match_pattern" file_1 file_2 file_3 ...
```

输出除之外的所有行 **-v** 选项：

```
grep -v "match_pattern" file_name
```

标记匹配颜色 **--color=auto** 选项：

```
grep "match_pattern" file_name --color=auto
```

使用正则表达式 **-E** 选项：

```
grep -E "[1-9]+"
# 或
egrep "[1-9]+"
```

使用正则表达式 **-P** 选项：

```
grep -P "(\d{3}\-){2}\d{4}" file_name
```

只输出文件中匹配到的部分 **-o** 选项：

```
echo this is a test line. | grep -o -E "[a-z]+\."
```

```
line.

echo this is a test line. | egrep -o "[a-z]+\."
line.
```

统计文件或者文本中包含匹配字符串的行数 **-c** 选项：

```
grep -c "text" file_name
```

输出包含匹配字符串的行数 **-n** 选项：

```
grep "text" -n file_name
#或
cat file_name | grep "text" -n

#多个文件
grep "text" -n file_1 file_2
```

打印样式匹配所位于的字符或字节偏移：

```
echo gun is not unix | grep -b -o "not"
7:not
#一行中字符串的字符便宜是从该行的第一个字符开始计算，起始值为0。选项 **-b -o** 一般总是配合使用。
```

搜索多个文件并查找匹配文本在哪些文件中：

```
grep -l "text" file1 file2 file3...
```

grep递归搜索文件

在多级目录中对文本进行递归搜索：

```
grep "text" . -r -n
#.表示当前目录。
```

忽略匹配样式中的字符大小写：

```
echo "hello world" | grep -i "HELLO"
# hello
```

选项 **-e** 制动多个匹配样式：

```
echo this is a text line | grep -e "is" -e "line" -o
is
line

#也可以使用 **-f** 选项来匹配多个样式，在样式文件中逐行写出需要匹配的字符。
cat patfile
aaa
bbb
```

```
echo aaa bbb ccc ddd eee | grep -f patfile -o
```

在grep搜索结果中包括或者排除指定文件：

```
# 只在目录中所有的.php和.html文件中递归搜索字符"main()"
grep "main()" . -r --include *.{php,html}

# 在搜索结果中排除所有README文件
grep "main()" . -r --exclude "README"

# 在搜索结果中排除filelist文件列表里的文件
grep "main()" . -r --exclude-from filelist
```

使用0值字节后缀的grep与xargs□

```
# 测试文件：
echo "aaa" > file1
echo "bbb" > file2
echo "aaa" > file3

grep "aaa" file* -lZ | xargs -0 rm

# 执行后会删除file1和file3□grep输出用-Z选项来指定以0值字节作为终结符文件名□\0□□xargs -0 读取输入并用0值字节终结符分隔文件名，然后删除匹配文件□-Z通常和-l结合使用。
```

grep静默输出：

```
grep -q "test" filename
# 不会输出任何信息，如果命令运行成功返回0，失败则返回非0值。一般用于条件测试。
```

打印出匹配文本之前或者之后的行：

```
# 显示匹配某个结果之后的3行，使用 -A 选项：
seq 10 | grep "5" -A 3
5
6
7
8

# 显示匹配某个结果之前的3行，使用 -B 选项：
seq 10 | grep "5" -B 3
2
3
4
5

# 显示匹配某个结果的前三行和后三行，使用 -C 选项：
seq 10 | grep "5" -C 3
2
3
```

```
4
5
6
7
8

# 如果匹配结果有多个，会用“ -- ”作为各匹配结果之间的分隔符：
echo -e "a\nb\nc\na\nb\nc" | grep a -A 1
a
b
--
a
b
```

grep查看某时间段的日志

```
#filename access.irust.top
irust.top 149.7.38.xx 30.001s - [10/Feb/2021:09:54:42 +0800] "POST /api/a
HTTP/1.1" 499 0 "-" - "SUP=- SUBP=-" "REQUEST_ID=-" "-" || Ali_Ecom14y110
irust.top 149.7.38.xx 30.002s - [10/Feb/2021:09:54:45 +0800] "POST /api/a
HTTP/1.1" 499 0 "-" - "SUP=- SUBP=-" "REQUEST_ID=-" "-" || Ali_Ecom13t186
irust.top 149.7.38.xx 30.000s - [10/Feb/2021:10:55:15 +0800] "POST /api/a
HTTP/1.1" 499 0 "-" - "SUP=- SUBP=-" "REQUEST_ID=-" "-" || Ali_Ecom10y138
irust.top 149.7.38.xx 30.002s - [10/Feb/2021:12:55:12 +0800] "POST /api/a
HTTP/1.1" 499 0 "-" - "SUP=- SUBP=-" "REQUEST_ID=-" "-" || Ali_Ecom15t140
....
```

```
grep '10/Feb/2021:0[0-9]' access.irust.top
```