



Cleiton Bueno

Sistemas Embarcados, Programação Multiplataforma, Linux
Embarcado e vivência open-source

Linux – Dominando o comando grep

O Linux possui muitos comandos nativos que são verdadeiros canivete-suiço para SysAdmins e criação de poderosos scripts.

Iniciei uma serie de posts sobre [Shell Script](#), e agora irei começar a utilizar diversos comandos do Linux para tornar e dar vida a grandes funções com estes scripts, para isso, começarei a abordar separadamente os mais conhecidos e usados.

Neste post vou falar sobre o grep, muitos não sabem mas o significado do nome é (Globally Search a Regular Expression and Print), o nome já diz tudo o que esse comando faz, e deixando mais claro a ideia é procurar texto em uma string ou dentro de arquivos e mostrar linhas, ocorrências, usar combinações para pesquisar e o resultado da pesquisa ser mostrado na tela.

Você não precisa se preocupar em instalar o grep, pois ele já está ae no seu Linux, para confirmar e verificar a versão veja o comando

abaixo:

```
1 $ grep
2 Usage: grep [OPTION]... PATTERN [FILE]...
3 Try 'grep --help' for more information.
4 $ grep -V
5 grep (GNU grep) 2.16
6 Copyright (C) 2014 Free Software Foundation, Inc.
7 License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl
8 This is free software: you are free to change and redistribute it.
9 There is NO WARRANTY, to the extent permitted by law.
10
11 Written by Mike Haertel and others, see <http://git.sv.gnu.org/cgit/gr
```

Digitando apenas o comando e dando enter ele não faz nada, porém mostra um exemplo para uso, e com o parâmetro -V a versão atual, no meu caso 2.6.

Agora segue os parâmetros que iremos utilizar e suas funções:

-c	Conta quantas vezes apareceu a string que esta pesquisando
-v	Mostra na tela "tudo" menos onde houver a ocorrência da string pesquisada
-i	Realiza uma busca pela string ignorando o case, sendo case-insensitive
-o	Ira mostrar na tela apenas as ocorrências da string pesquisada ignorando o resto
-n	Ira mostrar na tela na primeira coluna a linha onde encontrou a string pesquisada
-B	Numero de linhas a serem impressas antes da linha que contem a string pesquisada [BEFORE]
-A	Numero de linhas a serem impressas na tela depois da encontrar a linha com a string [AFTER]
-C	Quantidade de linhas antes e depois da linha que contem a string [CONTEXT]
-q	Ira procurar pela string informada, porém estará em modo silencioso, nada sera impresso na tela, porém caso encontre o comando encerra com 0, caso não encontre nada será 1
-E	Extende o uso de Regex no padrão e combinação, usando logica AND e OR por exemplo
-f	Um arquivo com combinações de padrões com Regex, podendo usar varias combinações

-l	Mostra somente o nome do arquivo onde foi encontrado a string pesquisada
-L	Semelhante ao -v, porque mostra apenas os arquivo que não contem a string informada
-h	Pesquisa varias arquivos, diretórios se com -r mas não mostra o nome dos arquivos
-r	Ira realizar uma pesquisa recursiva em todos os diretórios a partir do informado
-color	Deve-se passar o parâmetro 'never' caso não queira que a saída marque com cor a string ou 'auto' e 'always' para operar conforme necessite. Pode mudar a cor alterando GREP_COLOR, GREP_COLORS no environment

Exemplo básico

Agora um exemplo básico e bem didático para uso do comando, para isso vou criar um arquivo chamado palavras.txt e inserir um texto dentro e brincar com o grep.

```
1 echo -ne "amor\ncasa\nCasa\nCASA\nRaspberryPI\nRaspberry PI\nRaspberry
```

Vamos visualizar nosso arquivo palavras.txt.

```
1 $ cat palavras.txt
2 amor
3 casa
4 Casa
5 CASA
6 RaspberryPI
7 Raspberry PI
8 Raspberry B PI
9 Arduino
10 arduino
11 ARDUINO
12 IDEArduino
13 Linux é o poder
14 Eu programo Python e você?
```

Agora vamos usar o grep e pesquisar pela string "Raspberry", podemos usar de duas maneiras com o cat um pipe e logo em seguida um grep ou diretamente com o comando, a primeira opção

é muito utilizada, porém você irá perder performance caso realizar pesquisa em muitos arquivo ou em um arquivo longo, veremos ainda neste post sobre isso.

```
1 $ cat palavras.txt | grep "Raspberry"
2 RaspberryPI
3 Raspberry PI
4 Raspberry B PI
5
6 $ grep "Raspberry" palavras.txt
7 RaspberryPI
8 Raspberry PI
9 Raspberry B PI
```

Caso eu queira contar o numero de ocorrências da string "Raspberry":

```
1 $ grep -c "Raspberry" palavras.txt
2 3
```

Se eu quiser ver tudo menos a string que contenham "Raspberry":

```
1 $ grep -v "Raspberry" palavras.txt
2 amor
3 casa
4 Casa
5 CASA
6 Arduino
7 arduino
8 ARDUINO
9 IDEArduino
10 Linux é o poder
11 Eu programo Python e você?
```

Agora quero pesquisar pela string "arduino".

```
1 $ grep "arduino" palavras.txt
2 arduino
```

Temos Arduino escrito de diversas maneiras, então vamos pedir para ser case-insensitive.

```
1 $ grep -i "arduino" palavras.txt
2 Arduino
3 arduino
4 ARDUINO
5 IDEArduino
```

E se no lugar de mostrar a linha inteira ou o que estiver junto eu mostrar apenas a string procurada.

```
1 $ grep -o "arduino" palavras.txt
2 arduino
3 $ grep -oi "arduino" palavras.txt
4 Arduino
5 arduino
6 ARDUINO
7 Arduino
8 $ grep -oi "Raspberry" palavras.txt
9 Raspberry
10 Raspberry
11 Raspberry
```

Se eu precisar saber o numero da linha onde foi encontrada a string.

```
1 $ grep -n "Raspberry" palavras.txt
2 5:RaspberryPI
3 6:Raspberry PI
4 7:Raspberry B PI
5 $ grep -n "duino" palavras.txt
6 8:Arduino
7 9:arduino
8 11:IDEArduino
```

Agora vamos pesquisar pela string "arduino" e obter também as 2 linhas antes da string encontrada.

```
1 $ grep "arduino" -B 2 palavras.txt
2 Raspberry B PI
3 Arduino
4 arduino
```

O mesmo podemos fazer obtém as linhas depois da linha com a string pesquisada.

```
1 $ grep "arduino" -A 2 palavras.txt
2 arduino
3 ARDUINO
4 IDEArduino
```

E podemos unir as duas opções, pegando e imprimindo linhas antes e depois da linha que contem a string pesquisada.

```
1 $ grep "arduino" -C 2 palavras.txt
2 Raspberry B PI
```

```
3 Arduino
4 arduino
5 ARDUINO
6 IDEArduino
```

Caso não queira mostrar nada na tela, só saber se teve sucesso ou não na pesquisa.

```
1 $ grep -q "arduino" palavras.txt
2 $ echo $?
3 0
4
5 $ grep -q "Beaglebone" palavras.txt
6 $ echo $?
7 1
```

Apenas reforçando no exemplo acima pesquisei a string "arduino" com o parâmetro -q (modo silencioso) e peguei a saída do ultimo comando executado com (echo \$?), logo em seguida pesquisei por "Beaglebone" como não existe a saída foi 1.

Exemplo intermediário

Agora vamos subir um nível e brincar com outros parâmetros. Desta vez vamos criar mais 2 arquivos sistema.txt e hardware.txt, e também copiar a saída do dmesg para dmesg.log e brincar com estes caras.

Preparando os arquivos:

```
1 $ echo -ne "Linux Ubuntu\nLinux Debian\nLinux Mint\nLinux CentOS\nRaspb
2
3 $ echo -ne "ARM 1176JZF\nARM Cortex-A7\nBCM2835\nBCM2836\nBeaglebone BL
4
5 $ dmesg > dmesg.log
```

Eu criei sistema.txt e hardware.txt com palavras aleatórias, você pode agregar mais palavras para realizar seus testes.

Agora eu quero pesquisar em qualquer arquivo e que contenha a string "Raspberry".

```
1 $ grep "Raspberry" *
2 palavras.txt:RaspberryPI
3 palavras.txt:Raspberry PI
4 palavras.txt:Raspberry B PI
5 sistema.txt:Yocto RaspberryPI
6 sistema.txt:Buildroot RaspberryPI
7
8 $ grep "Raspberry" ./*
9 ./palavras.txt:RaspberryPI
10 ./palavras.txt:Raspberry PI
11 ./palavras.txt:Raspberry B PI
12 ./sistema.txt:Yocto RaspberryPI
13 ./sistema.txt:Buildroot RaspberryPI
```

Os demais parâmetros anteriores se aplicam aqui também.

```
1 $ grep -n "Raspberry" *
2 palavras.txt:5:RaspberryPI
3 palavras.txt:6:Raspberry PI
4 palavras.txt:7:Raspberry B PI
5 sistema.txt:6:Yocto RaspberryPI
6 sistema.txt:7:Buildroot RaspberryPI
```

Agora se eu criar um diretório exemplo/ e mover o palavras.txt para ele será que vai encontrar a string "Raspberry" nele ainda?

```
1 $ mkdir exemplo && mv palavras.txt exemplo/
2 $ grep "Raspberry" *
3 grep: exemplo: Is a directory
4 sistema.txt:Yocto RaspberryPI
5 sistema.txt:Buildroot RaspberryPI
```

Ele avisa que existe um diretório onde esta sendo feita a pesquisa, para que ele acesse o(s) diretório(s) deve-se passar o parâmetro -r para recursividade.

```
1 $ grep -r "Raspberry" *
2 exemplo/palavras.txt:RaspberryPI
3 exemplo/palavras.txt:Raspberry PI
4 exemplo/palavras.txt:Raspberry B PI
5 sistema.txt:Yocto RaspberryPI
6 sistema.txt:Buildroot RaspberryPI
```

As vezes só interessa saber a ocorrências mas não o arquivo.

```
1 $ grep -hr "Raspberry" *
```

Se caso queira apenas saber qual arquivo contem a string mas não precisa mostrar ela.

```
1 $ grep -lr "Raspberry" *  
2 exemplo/palavras.txt  
3 sistema.txt
```

E se quiser saber os arquivos que não possuem a string pesquisada.

```
1 $ grep -Lr "Raspberry" *  
2 dmesg.log  
3 hardware.txt
```

Habilitando ou não o uso da saída colorida.

```
1 $ grep -r --color=always "Raspberry" *  
2 exemplo/palavras.txt:RaspberryPI  
3 exemplo/palavras.txt:Raspberry PI  
4 exemplo/palavras.txt:Raspberry B PI  
5 sistema.txt:Yocto RaspberryPI  
6 sistema.txt:Buildroot RaspberryPI  
7  
8 $ grep -r --color=never "Raspberry" *  
9 exemplo/palavras.txt:RaspberryPI  
10 exemplo/palavras.txt:Raspberry PI  
11 exemplo/palavras.txt:Raspberry B PI  
12 sistema.txt:Yocto RaspberryPI  
13 sistema.txt:Buildroot RaspberryPI
```

Exemplo avançado

Vamos dar mais um passo sobre esse incrível comando, só que agora com o básico de Expressões Regulares, e como exemplo usaremos o dmesg.log gerado acima.

Fazendo uma busca simples pela string "usb".


```

1 $ grep "usb" dmesg.log
2 [ 0.668550] usbcore: registered new interface driver usbfs
3 [ 0.668558] usbcore: registered new interface driver hub
4 [ 0.668582] usbcore: registered new device driver usb
5 [ 1.996732] usb usb1: New USB device found, idVendor=1d6b, idProduct
6 [ 1.996735] usb usb1: New USB device strings: Mfr=3, Product=2, Ser
7 [ 1.996737] usb usb1: Product: EHCI Host Controller
8 [ 1.996739] usb usb1: Manufacturer: Linux 3.13.0-37-generic ehci_hc
9 [ 1.996741] usb usb1: SerialNumber: 0000:00:1d.0
10 [ 1.997338] usb usb2: New USB device found, idVendor=1d6b, idProduct
11 [ 1.997340] usb usb2: New USB device strings: Mfr=3, Product=2, Ser
12 [ 1.997342] usb usb2: Product: xHCI Host Controller
13 [ 1.997344] usb usb2: Manufacturer: Linux 3.13.0-37-generic xhci_hc
14 [ 1.997346] usb usb2: SerialNumber: 0000:00:14.0
15 [ 2.000099] usb usb3: New USB device found, idVendor=1d6b, idProduct
16 [ 2.000101] usb usb3: New USB device strings: Mfr=3, Product=2, Ser
17 [ 2.000103] usb usb3: Product: xHCI Host Controller
18 [ 2.000105] usb usb3: Manufacturer: Linux 3.13.0-37-generic xhci_hc
19 [ 2.000107] usb usb3: SerialNumber: 0000:00:14.0
20 [ 2.308561] usb 1-1: new high-speed USB device number 2 using ehci-
21 [ 2.440791] usb 1-1: New USB device found, idVendor=8087, idProduct
22 [ 2.440794] usb 1-1: New USB device strings: Mfr=0, Product=0, Seri
23 [ 2.712387] usb 1-1.5: new full-speed USB device number 3 using ehc
24 [ 2.805614] usb 1-1.5: New USB device found, idVendor=0cf3, idProdu
25 [ 2.805616] usb 1-1.5: New USB device strings: Mfr=0, Product=0, Se
26 [ 2.880293] usb 1-1.7: new high-speed USB device number 4 using ehc
27 [ 2.972951] usb 1-1.7: New USB device found, idVendor=0bda, idProdu
28 [ 2.972954] usb 1-1.7: New USB device strings: Mfr=1, Product=2, Se
29 [ 2.972956] usb 1-1.7: Product: USB2.0-CRW
30 [ 2.972958] usb 1-1.7: Manufacturer: Generic
31 [ 2.972959] usb 1-1.7: SerialNumber: 20100201396000000
32 [ 3.044205] usb 1-1.8: new high-speed USB device number 5 using ehc
33 [ 3.201201] usb 1-1.8: New USB device found, idVendor=0c45, idProdu
34 [ 3.201203] usb 1-1.8: New USB device strings: Mfr=2, Product=1, Se
35 [ 3.201205] usb 1-1.8: Product: Laptop_Integrated_Webcam_HD
36 [ 3.201206] usb 1-1.8: Manufacturer: CN0Y3PX8724873AGB17FA01
37 [ 14.243360] usbcore: registered new interface driver btusb
38 [ 14.274681] usbcore: registered new interface driver rts5139
39 [ 14.624063] input: Laptop_Integrated_Webcam_HD as /devices/pci0000:
40 [ 14.624169] usbcore: registered new interface driver uvcvideo
41 [ 14.761434] usbcore: registered new interface driver ath3k
42 [ 14.781788] usb 1-1.5: USB disconnect, device number 3
43 [ 14.981529] usb 1-1.5: new full-speed USB device number 6 using ehc
44 [ 20.075906] usb 1-1.5: New USB device found, idVendor=0cf3, idProdu
45 [ 20.075911] usb 1-1.5: New USB device strings: Mfr=0, Product=0, Se

```

Bastante coisa não é? Vamos trabalhar em cima extendendo os recursos de Regex da nossa expressão, por exemplo quero somente as linhas que contenham usb2 OU usb3, aplicando a logica OR.

```

1 $ grep -E "usb2|usb3" dmesg.log
2 [ 1.997338] usb usb2: New USB device found, idVendor=1d6b, idProduct
3 [ 1.997340] usb usb2: New USB device strings: Mfr=3, Product=2, Ser
4 [ 1.997342] usb usb2: Product: xHCI Host Controller
5 [ 1.997344] usb usb2: Manufacturer: Linux 3.13.0-37-generic xhci_hc
6 [ 1.997346] usb usb2: SerialNumber: 0000:00:14.0
7 [ 2.000099] usb usb3: New USB device found, idVendor=1d6b, idProduct

```

```
8 [ 2.000101] usb usb3: New USB device strings: Mfr=3, Product=2, Ser
9 [ 2.000103] usb usb3: Product: xHCI Host Controller
10 [ 2.000105] usb usb3: Manufacturer: Linux 3.13.0-37-generic xhci_hc
11 [ 2.000107] usb usb3: SerialNumber: 0000:00:14.0
```

Agora, vamos pesquisar por uma linha que contenha "usb" E também "Product:", vamos aplicar a logica AND.

```
1 $ grep -E "usb.*Product:" dmesg.log
2 [ 1.996737] usb usb1: Product: EHCI Host Controller
3 [ 1.997342] usb usb2: Product: xHCI Host Controller
4 [ 2.000103] usb usb3: Product: xHCI Host Controller
5 [ 2.972956] usb 1-1.7: Product: USB2.0-CRW
6 [ 3.201205] usb 1-1.8: Product: Laptop_Integrated_Webcam_HD
```

Mas eu quero só com "usb2" ou "usb3" casando com "Product:".

```
1 $ grep -E "usb(2|3).*Product:" dmesg.log
2 [ 1.997342] usb usb2: Product: xHCI Host Controller
3 [ 2.000103] usb usb3: Product: xHCI Host Controller
```

Sentiu o poder da ferramenta? E você pode aplicar varias combinações com Expressões Regulares, estudaremos no futuro sobre isso.

Podemos criar um arquivo com nosso Regex e usar ele como padrão, alias, podemos colocar varias combinações neste arquivo.

```
1 $ echo "usb(2|3).*Product:" > meu_regex
2 $ cat meu_regex
3 usb(2|3).*Product:
4 $ grep -f meu_regex -E dmesg.log
5 [ 1.997342] usb usb2: Product: xHCI Host Controller
6 [ 2.000103] usb usb3: Product: xHCI Host Controller
```

Usando direto e com pipe

Lembra quando comentei de usar o grep direto e usar ele com qualquer comando um pipe e logo em seguida o grep? Vamos fazer uma pesquisa no /var/log/syslog.1 por quantas ocorrências da

string "info" usando cat e o grep direto.

```

1 $ time cat /var/log/syslog.1 | grep -c "info"
2 1027
3
4 real    0m0.011s
5 user    0m0.005s
6 sys 0m0.008s
7
8
9 $ time grep -c "info" /var/log/syslog.1
10 1027
11
12 real    0m0.009s
13 user    0m0.009s
14 sys 0m0.000s

```

Agora uma dica legal, para desempenho de uso em grande escala do grep é setar LC_ALL=C antes:

```

1 $ strace -c grep -c "info" /var/log/syslog.1
2 187
3 % time      seconds  usecs/call   calls   errors syscall
4
5 -----
6 27.03      0.000160      11       14      mmap
7 14.19      0.000084       8       10      read
8 13.18      0.000078      13        6      open
9 11.15      0.000066       8        8      mprotect
10  9.63      0.000057      11        5      5 access
11  5.57      0.000033       4        9      fstat
12  4.56      0.000027       3        9      close
13  3.55      0.000021      11        2      munmap
14  3.55      0.000021       7        3      brk
15  3.04      0.000018      18        1      execve
16  1.52      0.000009       9        1      write
17  1.35      0.000008       4        2      1 ioctl
18  1.35      0.000008       8        1      openat
19  0.34      0.000002       2        1      arch_prctl
20 -----
21 100.00      0.000592      72       6 total
22
23 $ export LC_ALL=C
24
25 $ strace -c grep -c "info" /var/log/syslog.1
26 187
27 % time      seconds  usecs/call   calls   errors syscall
28
29 -----
30 20.07      0.000114      11       10      read
31 16.73      0.000095       8       12      mmap
32 16.20      0.000092      12        8      mprotect
33  6.87      0.000039       8        5      5 access
34  6.16      0.000035       9        4      open
35  6.16      0.000035      18        2      munmap
36  5.99      0.000034      34        1      execve
37  5.46      0.000031      10        3      brk
38  4.40      0.000025       4        7      close

```

39	3.87	0.000022	3	7	fstat
40	2.99	0.000017	17	1	write
41	2.64	0.000015	15	1	openat
42	1.94	0.000011	6	2	1 ioctl
43	0.53	0.000003	3	1	arch_prctl
44	-----				
45	100.00	0.000568		64	6 total

Novamente verificando o tempo apos setar LC_ALL=C.

```

1 $ time grep -c "info" /var/log/syslog.1
2 187
3
4 real    0m0.003s
5 user    0m0.003s
6 sys 0m0.000s
7
8
9 $ time cat /var/log/syslog.1 | grep -c "info"
10 187
11
12 real    0m0.005s
13 user    0m0.000s
14 sys 0m0.009s

```

No meu syslog.1 não deu tanta diferença, mas fazer essa varredura em um arquivo de 500M você notara uma grande diferença.

Conhecemos, aprendemos e devoramos o grep, vimos o grande potencial que ele pode nos oferecer, ainda possui mais recursos e parâmetros que não vimos e pode ser visto com `man grep`, mas o foco e suas principais funções foi exemplificado aqui.

Espero que tenham gostado e até a próxima!

Compartilhar



Linux – Dominando o comando grep by Cleiton Bueno is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tagged as: [grep](#), [linux grep](#), [regex](#)

Categorized in: [Linux](#)

 22 de julho de 2015

 [cleitonbueno](#)

 No Comments

14 Comments

www.cleitonbueno.com

 Login ▾ Recommend Tweet Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS Name **Marcio Barbosa** • a year ago

Saudações.

Sou novo nos comandos Linux, gostaria de saber se consigo encontrar palavras dentro de arquivos .docx ou .xlsx com esse comando. Se não for possível, qual comando devo utilizar?

Marcio

UBUNTU 18.1

 |  • Reply • Share ▸**Jesus Vieira de Lima** • 2 years ago


ID	Name	Public IPv4	Private IPv4	Public IPv6	Memory	VCPUS	Disk	Region	Image	Status	Tags
91198180	teste1				1024	1	30	nyc1	Ubuntu 16.04.4 x64	new	

Olá, estou com uma dúvida,

Veja este exemplo na imagem em anexo,

Eu gostaria de pegar o valor que está em baixo do ID e colocar em uma variável, você sabe como fazer? Poderia ajudar?

Obrigado pela atenção

 |  • Reply • Share ▸**Cleiton Bueno** Mod  Jesus Vieira de Lima • 2 years ago

Olá Jesus, vamos supor que o filtro seja a palavra chave "teste1" para pegar a linha, então depois disso você pode cortar a string separando por "<space>" e pegar o primeiro valor. Supondo que o arquivo a ser analisado chama-se meu_valor.txt, ficaria:

