



Os 40 Comandos Linux Mais Usados que Você Precisa Conhecer

A partir do momento em que escrevemos isto, o Linux tem uma [participação de mercado mundial de 2,68%](#) em desktops, mas mais de 90% de toda a infraestrutura de nuvem e [serviços de hospedagem](#) são executados neste sistema operacional. Somente por esta razão, é crucial estar familiarizado com os comandos populares do Linux.

De acordo com a [pesquisa StackOverflow 2020](#), Linux é o sistema operacional mais utilizado pelos desenvolvedores profissionais, com uma impressionante participação de 55,9% do mercado. Não é apenas uma coincidência. O Linux é gratuito e de código aberto, tem melhor segurança do que seus concorrentes e possui uma poderosa linha de comando que torna os desenvolvedores e os usuários poderosos mais eficientes. Você também tem acesso a um poderoso gerenciador de pacotes e um monte de ferramentas de desenvolvimento como o [DevKinsta](#).

Seja você um Sysadmin experiente ou um iniciante no Linux, você pode tirar proveito deste guia.

Vamos começar!

O que é um comando Linux?

Um comando Linux é um programa ou utilitário que roda na linha de comando. Uma [linha de comando](#) é uma interface que aceita linhas de texto e as processa em instruções para o seu computador.

Qualquer interface gráfica de usuário (GUI) é apenas uma abstração dos programas de linha de comando. Por exemplo, quando você fecha uma janela clicando no “X”, há um comando rodando atrás dessa ação.

Uma **flag** é uma forma de passarmos opções para o comando que você executa. A maioria dos comandos Linux tem uma página de ajuda que nós podemos chamar com a flag `-h`. Na maioria das vezes, as flags são opcionais.

Um **argument** ou parâmetro é a **input** que damos a um comando para que ele possa funcionar corretamente. Na maioria dos casos, o argument é um caminho de arquivo, mas ele pode ser qualquer coisa que você digitar no terminal.

Você pode invocar flags usando hífen(-) e hífen duplos(--), enquanto a execução do argument depende da ordem na qual você as passa para a função.

Os comandos Linux mais usados

Antes de pular para os comandos Linux mais utilizados, certifique-se de abri um **terminal**. Na maioria das distribuições Linux, você usaria `Ctrl + Alt + T` para fazer isso. Se isto não funcionar, procure em seu painel do aplicativo por “terminal”.



— O emulador de terminal Linux

Agora vamos mergulhar nos 40 comandos Linux mais utilizados. Muitos deles têm múltiplas opções que você pode utilizar, então certifique-se de [verificar o manual de comandos](#).

1. Comando `ls`

`ls` é provavelmente o primeiro comando que cada usuário do Linux digita em seu terminal. Ele permite que você liste o conteúdo do diretório que você quer (o diretório atual por

padrão), incluindo arquivos e outros diretórios aninhados.

```
ls
```

Ela tem muitas opções, então pode ser bom obter alguma ajuda usando a flag `--help`. Esta flag retorna todas as flags que você pode usar com `ls`.

Por exemplo, para colorir a saída do comando `ls`, você pode usar o seguinte:

```
ls --color=auto
```

```
~/Documents/linux-commands via 🐛 v3.9.6  
> ls --color=auto  
binarysearch.py  commands  dir1  dir2  dummyfile1.txt  get_keys.py  github_automation.py  important_file.txt  
~/Documents/linux-commands via 🐛 v3.9.6  
> |
```

— O comando colorido `ls`

Agora o comando `ls` está colorido, e você pode apreciar a diferença entre um diretório e um arquivo.

Mas digitar `ls` com a flag colorida seria ineficiente; é por isso que nós usamos o comando `alias`.

2. Comando `alias`

O comando `alias` permite que você defina alias temporários em sua sessão shell. Ao criar um alias, você instrui seu shell a substituir uma palavra por uma série de comandos.

Por exemplo, para definir `ls` para ter cor sem digitar a flag `--color` sempre, você usaria:

```
alias ls="ls --color=auto"
```

Como você pode ver, o comando `alias` toma um parâmetro de par de valores chave: `alias NAME="VALUE"`. Note que o valor deve estar dentro de aspas.

Se você quiser listar todos os apelidos que você tem em sua sessão shell, você pode executar o comando de `alias` sem argumentos.

```
alias
```

```
~/Documents/linux-commands via 🐉 v3.9.6
> alias
alias .. 'cd ..'
alias ... 'cd ../../'
alias 3.. 'cd ../../..'
alias awesome_server 'Xephyr -br -ac -noreset -screen 1300x730 :1 & DISPLAY=:1 awesome ~/.config/aweso
me/rc.lua'
alias cdC 'cd ~/.config/'
alias cdM 'cd ~/MEGAsync/'
alias cdMG 'cd ~/MEGAsync/github'
alias cl clear
alias config '/usr/bin/git --git-dir=/home/daniel/dotfiles/ --work-tree=/home/daniel'
alias config-a 'config add'
alias config-m 'config commit -m'
alias config-p 'config push origin'
alias config-s 'config status'
alias cp 'cp -r'
alias em '/usr/bin/emacs -nw'
alias emacs emacsclient\\| -c\\| -a\\| \\|'emacs\\|'
alias fish_key_reader /usr/bin/fish_key_reader
alias g git
alias gc 'git clone'
```

— O comando alias

3. Comando `unalias`

Como o nome sugere, o comando `unalias` visa remover um alias dos pseudônimos já definidos. Para remover o apelido `ls` anterior, você pode usar:

```
unalias ls
```

4. Comando `pwd`

O comando `pwd` significa “print working directory” (imprimir diretório de trabalho), e ele produz o caminho absoluto do diretório em que você está. Por exemplo, se o seu nome de usuário é “john” e você está em seu diretório de Documentos, seu caminho absoluto seria: `/home/john/Documents`.

Para usá-lo, simplesmente digite `pwd` no terminal:

```
pwd  
  
# My result: /home/kinsta/Documents/Documents/linux-commands
```

5. Comando `cd`

O comando `cd` é altamente popular, junto com o `ls`. Ele se refere a “change directory” e, como o nome sugere, muda você para o diretório que você está tentando acessar.

Por exemplo, se você está dentro do seu diretório de Documentos e está tentando acessar uma de suas subpastas chamada **Videos**, você pode digitá-la:

```
cd Videos
```

Você também pode fornecer o caminho absoluto da pasta:

```
cd /home/kinsta/Documents/Videos
```

Há alguns truques com o comando `cd` que podem economizar muito tempo quando se brinca com ele:

1. Ir para a pasta home

```
cd
```

2. Subir um nível

```
cd ..
```

3. Voltar para o diretório anterior

```
cd -
```

6. Comando `cp`

É tão fácil de copiar arquivos e pastas diretamente no terminal Linux que às vezes ele pode substituir os gerenciadores de arquivos convencionais.

Para usar o comando `cp`, basta digitá-lo junto com os arquivos de origem e destino:


```
cp file_to_copy.txt new_file.txt
```

Você também pode copiar diretórios inteiros usando a flag recursiva:

```
cp -r dir_to_copy/ new_copy_dir/
```

Lembre-se de que no Linux, as pastas terminam com uma barra (/).

7. Comando `rm`

Agora que você sabe como copiar arquivos, será útil saber como removê-los.

Você pode usar o comando `rm` para remover arquivos e diretórios. Mas tenha cuidado ao usá-lo, pois é muito difícil (mas não impossível) recuperar arquivos excluídos dessa forma.

Para excluir um arquivo normal, você digitaria:

```
rm file_to_copy.txt
```

Se você quiser apagar um diretório vazio, você pode usar a flag recursiva (`-r`):

```
rm -r dir_to_remove/
```

Por outro lado, para remover um diretório com conteúdo dentro dele, você precisa usar a força (-f) e flags recursivas:

```
rm -rf dir_with_content_to_remove/
```

Info

Tenha cuidado com isso – você pode apagar um dia inteiro de trabalho usando mal essas duas flags!

8. Comando `mv`

Você usa o comando `mv` para mover (ou renomear) arquivos e diretórios através do seu sistema de arquivos.

Para usar este comando, você digitaria seu nome com os arquivos de origem e destino:

```
mv source_file destination_folder/  
  
mv command_list.txt commands/
```

Para utilizar caminhos absolutos, você usaria:

```
mv /home/kinsta/BestMoviesOfAllTime ./
```

...onde ./ é o diretório onde você está atualmente.

Você também pode usar o `mv` para renomear arquivos enquanto os mantém no mesmo diretório:

```
mv old_file.txt new_named_file.txt
```

9. Comando `mkdir`

Para criar pastas na shell, você usa o comando `mkdir`. Basta especificar o nome da nova pasta, garantir que ela não existe e que você está pronto para ir.

Por exemplo, para fazer um diretório para manter [todas as suas imagens](#), basta digitar:

```
mkdir images/
```

Para criar subdiretórios com um simples comando, use o parent flag (-p):

```
mkdir -p movies/2004/
```

10. Comando `man`

Outro comando essencial do Linux é o `man`. Ele exibe a página de manual de qualquer outro comando (desde que ele tenha um).

Para ver a página do manual do comando `mkdir`, digite:

```
man mkdir
```

Você poderia até mesmo consultar a página de manual `man`:

```
man man
```

```

MAN(1)                               Manual pager utils                               MAN(1)

NAME
  man - an interface to the system reference manuals

SYNOPSIS
  man [man options] [[section] page ...] ...
  man -k [apropos options] regexp ...
  man -K [man options] [section] term ...
  man -f [whatis options] page ...
  man -l [man options] file ...
  man -w|-W [man options] page ...

DESCRIPTION
  man is the system's manual pager. Each page argument
  given to man is normally the name of a program, utility
  or function. The manual page associated with each of
  these arguments is then found and displayed. A section,
  if provided, will direct man to look only in that sec-
tion of the manual. The default action is to search in
  all of the available sections following a pre-defined
  order (see DEFAULTS), and to show only the first page
  found, even if page exists in several sections.

  The table below shows the section numbers of the manual
  followed by the types of pages they contain.

  1 Executable programs or shell commands
  2 System calls (functions provided by the kernel)
  3 Library calls (functions within program libraries)
  4 Special files (usually found in /dev)
  5 File formats and conventions, e.g. /etc/passwd
  6 Games

Manual page man(1) line 1 (press h for help or q to quit)

```

— A página do manual “man”

11. Comando touch

O comando `touch` permite a você atualizar os tempos de acesso e modificação dos arquivos especificados.

Por exemplo, eu tenho um arquivo antigo que foi modificado pela última vez em 12 de abril:

```
~/Documents/linux-commands via 🐍 v3.9.6
```

```
> ls -lah
```

Permissions	Size	User	Date	Modified	Name
drwxr-xr-x	-	daniel	8 ago	15:11	.
drwxr-xr-x	-	daniel	8 ago	00:27	..
drwxr-xr-x	-	daniel	8 ago	00:34	commands
drwxr-xr-x	-	daniel	7 ago	00:45	dir1
drwxr-xr-x	-	daniel	7 ago	00:45	dir2
drwxr-xr-x	-	daniel	8 ago	00:10	dir_to_copy
drwxr-xr-x	-	daniel	8 ago	00:12	new_dir
.rw-r--r--	0	daniel	8 ago	00:38	BestMoviesOfAllTime
.rw-r--r--	0	daniel	7 ago	00:44	binarysearch.py
.rw-r--r--	0	daniel	7 ago	00:43	dummyfile1.txt
.rw-r--r--	0	daniel	8 ago	00:18	file_to_delete.txt
.rw-r--r--	0	daniel	7 ago	00:44	get_keys.py
.rw-r--r--	0	daniel	7 ago	00:44	github_automation.py
.rw-r--r--	0	daniel	7 ago	00:44	important_file.txt
.rw-r--r--	0	daniel	8 ago	00:04	new_file.txt
.rw-r--r--	0	daniel	12 abr	20:45	old_file

— Data antiga

Para mudar sua data de modificação para a hora atual, nós precisamos usar a flag `-m`:

```
touch -m old_file
```

Agora a data corresponde à data de hoje (que no momento em que escrevemos foi 8 de agosto).

```
~/Documents/linux-commands via 🐍 v3.9.6
> ls -lah
Permissions Size User Date Modified Name
drwxr-xr-x - daniel 8 ago 15:11 .
drwxr-xr-x - daniel 8 ago 00:27 ..
drwxr-xr-x - daniel 8 ago 00:34 commands
drwxr-xr-x - daniel 7 ago 00:45 dir1
drwxr-xr-x - daniel 7 ago 00:45 dir2
drwxr-xr-x - daniel 8 ago 00:10 dir_to_copy
drwxr-xr-x - daniel 8 ago 00:12 new_dir
.rw-r--r-- 0 daniel 8 ago 00:38 BestMoviesOfAllTime
.rw-r--r-- 0 daniel 7 ago 00:44 binarysearch.py
.rw-r--r-- 0 daniel 7 ago 00:43 dummyfile1.txt
.rw-r--r-- 0 daniel 8 ago 00:18 file_to_delete.txt
.rw-r--r-- 0 daniel 7 ago 00:44 get_keys.py
.rw-r--r-- 0 daniel 7 ago 00:44 github_automation.py
.rw-r--r-- 0 daniel 7 ago 00:44 important_file.txt
.rw-r--r-- 0 daniel 8 ago 00:04 new_file.txt
.rw-r--r-- 0 daniel 8 ago 16:30 old_file
```

— Nova data

No entanto, na maioria das vezes, você não vai usar `touch` para modificar datas de arquivos, mas sim para criar novos arquivos vazios:

```
touch new_file_name
```

12. Comando `chmod`

O comando `chmod` permite que você mude o [modo de um arquivo](#) (permissões) rapidamente. Ele tem um monte de opções disponíveis com ele.

As permissões básicas que um arquivo pode ter são:

- r (read)
- w (write)
- x (execute)

Um dos casos mais comuns de uso do `chmod` é tornar um arquivo executável pelo usuário. Para fazer isso, digite `chmod` e a flag `+x`, seguida do arquivo no qual você quer modificar as permissões:

```
chmod +x script
```

Você usa isso para tornar os scripts executáveis, permitindo que você os execute diretamente, usando a notação `./`.

13. Comando `./`

Talvez a notação `./` não seja um comando em si, mas vale a pena mencionar nesta lista. Ele permite que seu shell execute um arquivo executável com qualquer interpretador instalado em seu sistema diretamente do terminal. Chega de fazer duplo clique em um arquivo em um gerenciador de arquivos gráfico!

Por exemplo, com este comando, você pode executar um [script Python](#) ou um programa disponível apenas no formato `.run`, como o [XAMPP](#). Ao executar um executável, certifique-se de que ele tenha permissões executáveis (x), que você pode modificar com o comando `chmod`.

Aqui está um simples script Python e como nós o executaríamos com a notação `./`:

```
#!/usr/bin/python3

# filename: script

for i in range(20):

    print(f"This is a cool script {i}")
```

Veja como nós converteríamos o script em um executável e o executaríamos:

```
chmod +x script

./script
```

14. Comando `exit`

O comando `exit` faz exatamente o que seu nome sugere: Com ele, você pode terminar uma sessão shell e, na maioria dos casos, fechar automaticamente [o terminal](#) que você está usando:

```
exit
```

15. Comando `sudo`

Este comando significa “superuser do”, e permite que você aja como um superusuário ou usuário root enquanto você estiver executando um comando específico. É como o Linux se protege e evita que os usuários modifiquem acidentalmente o sistema de arquivos da máquina ou instalem pacotes inadequados.

O Sudo é comumente usado para instalar software ou para editar arquivos fora do diretório home do usuário:

```
sudo apt install gimp  
sudo cd /root/
```

Ele irá pedir a senha do administrador antes de executar o comando que você digitou após ele.

16. Comando `shutdown`

Como você pode adivinhar, o comando de `shutdown` permite que você desligue sua máquina. No entanto, ele também pode ser usado para pará-lo e reiniciá-lo.

Para desligar o seu computador imediatamente (o padrão é um minuto), digite:

```
shutdown now
```

Você também pode programar para desligar o seu sistema em um formato de 24 horas:

```
shutdown 20:40
```

Para cancelar uma chamada `shutdown` anterior, você pode usar a flag `-c`:

```
shutdown -c
```

17. Comando `htop`

`htop` é um visualizador de processos interativo que permite a você gerenciar os recursos da sua máquina diretamente do terminal. Na maioria dos casos, ele não é instalado por padrão, então certifique-se de ler mais sobre ele [em sua página de download](#).

```
htop
```

```

0[ | 1.3% | 3[ | 1.3% | 6[ | 2.6% | 9[ | 4.6%
1[ | 2.0% | 4[ | 2.0% | 7[ | 2.6% | 10[ | 3.3%
2[ | 0.7% | 5[ | 4.7% | 8[ | 0.7% | 11[ | 7.8%
Mem[ | 5.77G/15.6G | Tasks: 114, 760 thr; 2 running
Swp[ | 0K/7.81G | Load average: 1.73 1.33 1.19
Uptime: 14:33:50

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
12648 daniel 20 0 41.8G 1275M 108M R 10.5 8.0 1h34:18 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
1123 daniel 20 0 398M 78540 21976 S 2.0 0.5 17:04.77 /usr/bin/python /usr/bin/qtile start
726 root 20 0 6283M 75536 36020 S 1.3 0.5 1h37:14 /usr/lib/Xorg :0 -seat seat0 -auth /run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
1239 daniel 9 -11 1001M 14408 9940 S 1.3 0.1 19:49.15 /usr/bin/pulseaudio --daemonize=no --log-target=journal
12657 daniel 20 0 41.8G 1275M 108M S 1.3 8.0 6:56.41 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
388163 daniel 20 0 529M 86128 40580 S 1.3 0.5 0:52.86 alacritty
1199 daniel 20 0 398M 78540 21976 S 0.7 0.5 0:27.02 /usr/bin/python /usr/bin/qtile start
4051 daniel 20 0 1856M 556M 172M S 0.7 3.5 16:47.11 /usr/lib/brave/brave
4096 daniel 20 0 1339M 437M 131M S 0.7 2.7 1h05:42 /usr/lib/brave/brave --type=gpu-process --field-trial-handle=1383049695608654963,16601138126615002718,1310
4101 daniel 20 0 398M 110M 68656 S 0.7 0.7 11:26.87 /usr/lib/brave/brave --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1
4134 daniel 20 0 1339M 437M 131M S 0.7 2.7 17:41.35 /usr/lib/brave/brave --type=gpu-process --field-trial-handle=1383049695608654963,16601138126615002718,1310
4383 daniel 20 0 37.3G 150M 90520 S 0.7 0.9 0:39.22 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
12651 daniel 20 0 41.8G 1275M 108M S 0.7 8.0 1:42.30 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
386997 daniel 20 0 37.3G 192M 94428 S 0.7 1.2 5:14.14 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
387087 daniel 20 0 37.3G 192M 94428 S 0.7 1.2 3:09.35 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
387210 daniel 20 0 37.4G 162M 98056 S 0.7 1.0 1:55.13 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
908280 daniel 20 0 37.6G 424M 307M S 0.7 2.7 1:06.45 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
962613 daniel 20 0 41.5G 211M 103M S 0.7 1.3 0:17.04 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
963864 daniel 20 0 343M 58608 32556 S 0.7 0.4 0:14.15 /usr/lib/brave/brave --type=utility --utility-sub-type=media.mojom.CdmService --field-trial-handle=1383049
1021985 daniel 20 0 37.3G 127M 97020 S 0.7 0.8 0:00.73 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
1023803 daniel 20 0 13864 7092 3500 R 0.7 0.0 0:00.41 htop
1 root 20 0 167M 11084 8228 S 0.0 0.1 0:00.80 /sbin/init
399 root 20 0 8164 4784 1648 S 0.0 0.0 0:03.77 /usr/bin/haveged -w 1024 -v 1 --Foreground
400 root 20 0 59944 25860 24660 S 0.0 0.2 0:00.30 /usr/lib/systemd/systemd-journald
401 root 20 0 31828 9720 6776 S 0.0 0.1 0:00.36 /usr/lib/systemd/systemd-udev
590 avahi 20 0 12720 5308 4620 S 0.0 0.0 0:00.08 avahi-daemon: running [danielmanjaro.local]
591 root 20 0 9200 2692 2344 S 0.0 0.0 0:00.02 /usr/bin/cron -n
592 dbus 20 0 13840 6720 5016 S 0.0 0.0 0:00.46 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
593 root 20 0 470M 19968 16980 S 0.0 0.1 0:00.86 /usr/bin/NetworkManager --no-daemon
595 polkitd 20 0 2914M 23612 18368 S 0.0 0.1 0:00.12 /usr/lib/polkit-1/polkitd --no-debug
601 root 20 0 175M 8184 6900 S 0.0 0.0 0:00.87 /usr/lib/systemd/systemd-logind
602 root 20 0 14716 6828 6032 S 0.0 0.0 0:00.07 /usr/lib/systemd/systemd-machined
613 avahi 20 0 12448 688 0 S 0.0 0.0 0:00.00 avahi-daemon: chroot helper
701 polkitd 20 0 2914M 23612 18368 S 0.0 0.1 0:00.00 /usr/lib/polkit-1/polkitd --no-debug

```

— A interface “htop”

18. Comando unzip

O comando [unzip](#) permite que você extraia o conteúdo de um arquivo **.zip** do terminal. Mais uma vez, este pacote pode não ser instalado por padrão, então certifique-se de instalá-lo com seu gerenciador de pacotes.

Aqui, estamos extraindo um arquivo **.zip** cheio de imagens:

```
unzip images.zip
```

19. Comandos apt, yum, pacman

Não importa qual distribuição Linux você esteja usando, é provável que você use gerenciadores de pacotes para instalar, atualizar e remover o software que você usa todos os dias.

Você pode acessar esses gerentes de pacotes através da linha de comando, e você usaria um ou outro dependendo da distro que sua máquina estiver rodando.

Os exemplos a seguir irão instalar o [GIMP](#), um software gratuito e de código aberto normalmente disponível na maioria dos gerentes de pacotes:

1. Baseado no Debian (Ubuntu, Linux Mint)

```
sudo apt install gimp
```

2. Baseado no Red Hat (Fedora, CentOS)

```
sudo yum install gimp
```

3. Baseado no Arco (Manjaro, Arco Linux)

```
sudo pacman -S gimp
```

20. Comando echo

O comando `echo` exibe o texto definido no terminal – é tão simples quanto isso:

```
echo "Cool message"
```

A terminal window with a dark background. The prompt is a tilde (~). The command `echo "Cool message"` is entered, with `echo` in blue and the string in yellow. The output `Cool message` is shown below the command.

```
~  
> echo "Cool message"  
Cool message
```

— O comando echo

Seu principal uso é imprimir variáveis ambientais dentro dessas mensagens:

```
echo "Hey $USER"  
  
# Hey kinsta
```

21. Comando cat

`Cat`, abreviação de “concatenate”, permite a você criar, visualizar e concatenar arquivos diretamente do terminal. É usado principalmente para pré-visualizar um arquivo sem abrir um editor de texto gráfico:

```
cat long_text_file.txt
```

```
~/Documents/linux-commands via 🐍 v3.9.6  
> cat long_text_file.txt  
Not that large at all! :)
```

— O comando cat

22. Comando `ps`

Com `ps`, você pode dar uma olhada nos processos que sua sessão de shell atual está rodando. Ele imprime informações úteis sobre os programas que você está executando, como ID do processo, TTY (TeleTYpewriter), tempo e nome do comando.

```
ps
```

```
~  
> ps  
    PID TTY          TIME CMD  
 533494 pts/2    00:00:00 fish  
 539315 pts/2    00:00:00 ps
```

— O comando ps

No caso de você querer algo mais interativo, você pode usar `htop`.

23. Comando `kill`

É irritante quando um programa não responde, e você não pode fechá-lo de forma alguma. Felizmente, o comando `kill` resolve este tipo de problema.

Simplificando, `kill` envia um `TERM` ou sinal de matar para um processo que o encerra.

Você pode matar processos digitando o PID (Process ID) ou o nome binário do programa:

```
kill 533494  
  
kill firefox
```

Tenha cuidado com este comando – com `kill`, você corre o risco de apagar acidentalmente o trabalho que você tem feito.

24. Comando `ping`

`ping` é o mais popular utilitário de terminal de rede usado para testar a conectividade de rede. `ping` tem uma tonelada de opções, mas na maioria dos casos, você vai usá-lo para solicitar um domínio ou [endereço IP](#):

```
ping google.com
```

```
ping 8.8.8.8
```

25. Comando `vim`

`vim` é um editor de texto de terminal gratuito e de código aberto que é usado desde os anos 90. Ele permite que você edite arquivos de texto simples usando eficientemente keybindings.

Algumas pessoas consideram difícil de usar – [sair de Vim](#) é uma das perguntas mais vistas do StackOverflow – mas uma vez que você se acostuma, ele se torna seu melhor aliado na linha de comando.

Para ligar o Vim, basta digitar:

```
vim
```



```
256 man type
257 kill firefox
258 cat old_file
259 ping google.com
260 ping 8.8.8.8
261 ping -c 8 google.com
262 ps
263 cd
264 ls
265 history
[daniel@danielmanjaro ~]$
```

— O comando history

27. Comando `passwd`

`passwd` permite que você [altere as senhas](#) das contas de usuário. Primeiro, ele solicita que você insira sua senha atual, depois pede uma nova senha e uma confirmação.

É semelhante a qualquer outra mudança de senha que você tenha visto em outro lugar, mas neste caso, está diretamente no seu terminal:

```
passwd
```

```
~  
> passwd  
Changing password for daniel.  
Current password: █
```

— O comando passwd

Tenha cuidado ao usá-la – você não quer bagunçar sua senha de usuário!

28. Comando `which`

O comando `which` produz o caminho completo dos comandos shell. Se ele não conseguir reconhecer o comando dado, ele vai lançar um erro.

Por exemplo, podemos usar isso para verificar o caminho binário para [Python](#) e o [navegador da web Brave](#):

```
which python  
  
# /usr/bin/python  
  
which brave  
  
# /usr/bin/brave
```

29. Comando `shred`

Se você já quis que um arquivo fosse quase impossível de [recuperar](#), `shred` pode ajudá-lo com esta tarefa. Este comando substitui o conteúdo de um arquivo repetidamente e, como

resultado, o arquivo dado torna-se extremamente difícil de ser recuperado.

Aqui está um arquivo com pouco conteúdo nele:

```
~/Documents/linux-commands via 🐧 v3.9.6  
> cat file_to_shred.txt  
A testing file, :))
```

— Arquivo para triturar

Agora, vamos fazer `shred` sua parte digitando o seguinte comando:

```
shred file_to_shred.txt
```



```
less large_text_file.txt
```

```
Hey, you should be using less more!  
Vim is the best editor, check it out.  
You can start with any programming language, but do it with Python.  
  
Hey, you should be using less more!  
Vim is the best editor, check it out.  
You can start with any programming language, but do it with Python.  
  
Hey, you should be using less more!  
Vim is the best editor, check it out.  
You can start with any programming language, but do it with Python.  
  
Hey, you should be using less more!  
Vim is the best editor, check it out.  
You can start with any programming language, but do it with Python.  
  
Hey, you should be using less more!  
Vim is the best editor, check it out.  
You can start with any programming language, but do it with Python.  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
(END)
```

— O comando less

O mais legal do `less` é que ele inclui mais comandos `vim` em sua interface. Se você precisa de algo mais interativo que o `cat`, `less` é uma boa opção.

31. Comando `tail`

Similar ao `cat`, o `tail` imprime o conteúdo de um arquivo com uma grande ressalva: ela só produz as últimas linhas. Por padrão, ele imprime as últimas 10 linhas, mas você pode modificar esse número com `-n`.

Por exemplo, para imprimir as últimas linhas de um arquivo de texto grande, você usaria:

```
tail long.txt
```

```
~/Documents/linux-commands via 🐧 v3.9.6 took 3m38s  
> tail long.txt  
Hey, we're almost there.  
These are the last lines of this text file.  
We're trying to test out some linux commands, and this is a good sample text to do it.  
To conclude, linux commands let you save a lot of time while being on a terminal or command line.  
This is the end of this file byee!!!
```

— O comando `tail`

Para ver apenas as últimas quatro linhas:

```
tail -n 4 long.txt
```



```
~/Documents/linux-commands via 🐉 v3.9.6  
> tail -n 4 long.txt
```

To conclude, linux commands let you save a lot of time while being on a terminal or command line.

```
This is the end of this file bye!!!!
```



— tail quatro linhas

32. Comando head

Este é complementar ao comando `tail`. O `head` produz as primeiras 10 linhas de um arquivo de texto, mas você pode definir qualquer número de linhas que você queira exibir com a flag `-n`:

```
head long.txt
```

```
head -n 5 long.txt
```

```
~/Documents/linux-commands via 🐙 v3.9.6
```

```
> head long.txt
```

```
Beggining of this large file!
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
~/Documents/linux-commands via 🐙 v3.9.6
```

```
> head -n 5 long.txt
```

```
Beggining of this large file!
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

```
Here goes a ton of content.
```

— O comando head

33. Comando grep

Grep é uma das mais poderosas utilidades para trabalhar com arquivos de texto. Ele procura por linhas que correspondam a uma [expressão regular](#) e as imprime:

```
grep "linux" long.txt
```

```
~/Documents/linux-commands via 🐧 v3.9.6
> grep "linux" long.txt
We're trying to test out some linux commands, and this is a good sample text to do it.
To conclude, linux commands let you save a lot of time while being on a terminal or command line.
```

— O comando grep

Você pode contar o número de vezes que o padrão se repete, usando a flag `-c`:

```
grep -c "linux" long.txt

# 2
```

34. Comando `whoami`

O comando `whoami` (abreviação para “who am i”) exibe o [nome de usuário](#) atualmente em uso:

```
whoami

# kinsta
```

Você obteria o mesmo resultado usando `echo` e a variável ambiental `$USER`:

```
echo $USER
```

```
# kinsta
```

35. Comando `what is`

`what is` imprime uma descrição de uma única linha de qualquer outro comando, tornando-a uma referência útil:

```
what is python
```

```
# python (1) - an interpreted, interactive, object-oriented programming language
```

```
what is what is
```

```
# what is (1) - display one-line manual page descriptions
```

36. Comando `wc`

`Wc` significa “word count”, e como o nome sugere, ele retorna o número de palavras em um arquivo de texto:

```
wc long.txt
```

```
# 37 207 1000 long.txt
```

Vamos quebrar a saída deste comando:

- 37 linhas
- 207 palavras
- Tamanho de 1000 bytes
- O nome do arquivo (long.txt)

Se você só precisa do número de palavras, use a flag `-w`:

```
wc -w long.txt  
  
207 long.txt
```

37. Comando `uname`

`uname`(abreviação de “Unix name”) imprime as informações operacionais do sistema, o que vem a calhar quando você conhece sua versão atual do Linux.

Na maioria das vezes, você estará usando a flag `-a` (`-all`), já que a saída padrão não é tão útil:

```
uname  
  
# Linux  
  
uname -a  
  
# Linux kinstamanjaro 5.4.138-1-MANJARO #1 SMP PREEMPT Thu Aug 5 12:15:21
```


Na maioria das máquinas, este comando não está disponível por padrão, então certifique-se de instalá-lo com seu gerenciador de pacotes primeiro.

39. Comando `find`

O comando `find` procura por [arquivos em uma hierarquia de diretórios](#) baseada em uma expressão regex. Para usá-lo, siga a sintaxe abaixo:

```
find [flags] [path] -name [expression]
```

Para procurar por um arquivo chamado **long.txt** no diretório atual, digite isto:

```
find ./ -name "long.txt" # ./long.txt
```

Para procurar por arquivos que terminam com uma extensão **.py** (Python), você pode usar o seguinte comando:

```
find ./ -type f -name "*.py" ./get_keys.py ./github_automation.py ./binary
```

40. Comando `wget`

`wget` (World Wide Web get) é um utilitário para recuperar conteúdo da internet. Ele tem uma das maiores coleções de flags por aí.

Aqui está como você baixaria um arquivo Python de um repo do [GitHub](#):

```
wget https://raw.githubusercontent.com/DaniDiazTech/Object-Oriented-Program
```

Resumo dos comandos do Linux

Sempre que você quiser uma referência rápida, basta rever a tabela abaixo:

Comando	Uso
ls	Lista o conteúdo de um diretório
alias	Define ou exhibe pseudônimos
unalias	Remove definições do <code>alias</code>
pwd	Imprime o diretório de trabalho
cd	Diretório de mudanças
cp	Copia arquivos e diretórios
rm	Remove arquivos e diretórios
mv	Movimenta (renomeia) arquivos e diretórios
mkdir	Cria diretórios
man	Exibe página de manual de outros comandos
touch	Cria arquivos vazios
chmod	Muda as permissões dos arquivos
./	Executa um executável
exit	Sai da atual sessão shell
sudo	Executa comandos como superusuário
shutdown	Desliga a sua máquina
htop	Exibe informações sobre processos e recursos
unzip	Extrai arquivos compactados ZIP
apt, yum, pacman	Gerenciadores de pacotes
echo	Exibe linhas de texto
cat	Imprime o conteúdo do arquivo
ps	Relata o status do processo shell
kill	Termina programas
ping	Testa a conectividade de rede
vim	Edição eficiente de textos
history	Mostra uma lista de comandos anteriores
passwd	Muda a senha do usuário
which	Retorna o caminho binário completo de um programa

Comando	Uso
shred	Sobreescreve um arquivo para esconder seu conteúdo
less	Inspeciona arquivos de forma interativa
tail	Exibe as últimas linhas de um arquivo
head	Exibe as primeiras linhas de um arquivo
grep	Imprime linhas que combinam padrões
whoami	Produz o nome de usuário
whatis	Mostra descrições de uma linha
wc	Arquivos de contagem de palavras
uname	Exibe informações do sistema operacional
neofetch	Exibe informações de sistema operacional e hardware
find	Busca por arquivos que seguem um padrão
wget	Recupera arquivos da internet

Resumo

Pode levar algum tempo para aprender Linux, mas uma vez que você domine algumas de suas ferramentas, ele se torna seu melhor aliado, e você não se arrepende de tê-lo escolhido como seu motorista diário.

Uma das coisas notáveis sobre o Linux é que mesmo que você seja um usuário experiente, você nunca vai parar de aprender a ser mais produtivo usando-o.

Há muito mais comandos Linux úteis. Se deixamos algo de fora, por favor, compartilhe seus comandos favoritos do Linux nos comentários abaixo!