



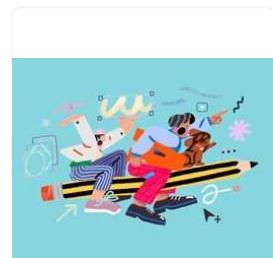
Flávio Copes

==

O Manual da Linha de Comando

Índice

- [Índice](#)
- [1. Introdução ao Linux](#)
- [2. homem](#)
- [3. ls](#)
- [4. cd](#)
- [5. pwd](#)
- [6. mkdir](#)
- [7. rmdir](#)
- [8. mv](#)
- [9. cp](#)
- [10. aberto](#)
- [11. toque](#)
- [12. encontrar](#)
- [13. ln](#)
- [14. gzip](#)
- [15. Gunzip](#)
- [16.tar](#)
- [17. pseudônimo](#)
- [18. gato](#)
- [19. menos](#)
- [20. cauda](#)
- [21. wc](#)
- [22. grep](#)
- [23. classificar](#)
- [24. uniq](#)



Adobe Creative
Cloud para equipes.
Coloque a
criatividade para
trabalhar.

ANÚNCIOS VIA CARBON

- [25. diff](#)
- [26. eco](#)
- [27. chown](#)
- [28. chmod](#)
- [29. umask](#)
- [30. du](#)
- [31. df](#)
- [32. nome base](#)
- [33. dirname](#)
- [34. ps](#)
- [35. topo](#)
- [36. matar](#)
- [37. matar](#)
- [38. empregos](#)
- [39. bg](#)
- [40. fg](#)
- [41. digite](#)
- [42. que](#)
- [43. nohup](#)
- [44. xargs](#)
- [45. vim](#)
- [46. emacs](#)
- [47. nano](#)
- [48. whoami](#)
- [49. quem](#)
- [50. su](#)
- [51. sudo](#)
- [52. passwd](#)
- [53. ping](#)
- [54. traceroute](#)
- [55. Claro](#)
- [56. história](#)
- [57. exportação](#)
- [58. crontab](#)
- [59. uname](#)
- [60. env](#)
- [61. impressão](#)

1. Introdução ao Linux

Linux é um sistema operacional, como macOS ou Windows.

É também o mais popular Open Source e sistema operacional livre, como em liberdade.

Ele alimenta a grande maioria dos servidores que compõem a Internet. É a base sobre a qual tudo é construído. Mas não é só isso. O Android é baseado em (uma versão modificada do) Linux.

O "núcleo" do Linux (chamado *kernel*) nasceu em 1991 na Finlândia, e percorreu um longo caminho desde o seu humilde início. Ele passou a ser o kernel do Sistema Operacional GNU, criando a dupla GNU/Linux.

Há uma coisa sobre o Linux que corporações como a Microsoft e a Apple, ou o Google, nunca serão capazes de oferecer: a liberdade de fazer o que quiser com o seu computador.

Eles estão realmente indo na direção oposta, construindo jardins murados, especialmente no lado móvel.

Linux é a liberdade final.

Ele é desenvolvido por voluntários, alguns pagos por empresas que dependem dele, alguns de forma independente, mas não há uma única empresa comercial que possa ditar o que entra no Linux ou as prioridades do projeto.

O Linux também pode ser usado como seu computador do dia a dia. Eu uso o macOS porque eu realmente gosto dos aplicativos, do design e eu também costumava ser um desenvolvedor de aplicativos iOS e Mac, mas antes de usá-lo eu usei o Linux como meu sistema operacional do computador principal.

Ninguém pode ditar quais aplicativos você pode executar ou "ligar para casa" com aplicativos que rastreiam você, sua posição e muito mais.

O Linux também é especial porque não há apenas "um Linux", como acontece no Windows ou no macOS. Em vez disso, temos **distribuições**.

Uma "distro" é feita por uma empresa ou organização e empacota o núcleo do Linux com programas e ferramentas adicionais.

Por exemplo, você tem o Debian, Red Hat e Ubuntu, provavelmente os mais populares.

Muitos, muitos mais existem. Você também pode criar sua própria distribuição. Mas o mais provável é que você use um popular, que tenha muitos usuários e uma comunidade de pessoas ao redor, para que você possa fazer o que precisa fazer sem perder muito tempo reinventando a roda e descobrindo respostas para problemas comuns.

Alguns computadores desktop e laptops são fornecidos com o Linux pré-instalado. Ou você pode instalá-lo em seu computador baseado no Windows ou em um Mac.

Mas você não precisa interromper seu computador existente apenas para ter uma ideia de como o Linux funciona.

Eu não tenho um computador Linux.

Se você usa um Mac, você precisa saber que sob o capô o macOS é um sistema operacional UNIX, e ele compartilha muitas das mesmas ideias e software que um sistema GNU/Linux usa, porque o GNU/Linux é uma alternativa livre ao UNIX.

UNIX é um termo guarda-chuva que agrupa muitos sistemas operacionais usados em grandes corporações e instituições, a partir da década de 70.

O terminal macOS dá-lhe acesso aos mesmos comandos exatos que descreverei no resto deste manual.

A Microsoft tem um subsistema oficial do Windows para Linux que você pode (e deve!) instalar no Windows. Isso lhe dará a capacidade de executar o Linux de uma maneira muito fácil no seu PC.

Mas na grande maioria das vezes você executará um computador Linux na nuvem através de um VPS (Virtual Private Server) como o DigitalOcean.

Um shell é um interpretador de comandos que expõe ao usuário uma interface para trabalhar com o sistema operacional subjacente.

Ele permite que você execute operações usando texto e comandos, e fornece aos usuários recursos avançados, como a capacidade de criar scripts.

Isso é importante: os shells permitem que você execute as coisas de uma maneira mais otimizada do que uma GUI (Interface Gráfica do Usuário) poderia permitir que você fizesse. As ferramentas de linha de comando podem oferecer muitas opções de configuração diferentes sem serem muito complexas de usar.

Existem muitos tipos diferentes de conchas. Este post se concentra em shells Unix, aqueles que você encontrará comumente em computadores Linux e macOS.

Muitos tipos diferentes de conchas foram criados para esses sistemas ao longo do tempo, e alguns deles dominam o espaço: Bash, Csh, Zsh, Fish e muitos mais!

Todos os shells se originam do Bourne Shell, chamado . "Bourne" porque seu criador foi Steve Bourne.sh

Bash significa *Bourne-again shell*. era proprietário e não de código aberto, e o Bash foi criado em 1989 para criar uma alternativa livre para o projeto GNU e a Free Software Foundation. Como os projetos tinham que pagar para usar a concha Bourne, o Bash tornou-se muito popular.sh

Se você usa um Mac, tente abrir o terminal do Mac. Que, por padrão, está executando o ZSH. (ou, pré-Catalina, Bash)

Você pode configurar seu sistema para executar qualquer tipo de concha, por exemplo, eu uso a concha de peixe.

Cada shell tem seus próprios recursos exclusivos e uso avançado, mas todos eles compartilham uma funcionalidade comum: eles podem permitir que você execute programas e eles podem ser programados.

No restante deste manual, veremos em detalhes os comandos mais comuns que você usará.

2. homem

O primeiro comando que quero introduzir é um comando que irá ajudá-lo a entender todos os outros comandos.

Toda vez que eu não sei como usar um comando, eu digito para obter o manual:`man <command>`

```
LS(1)          BSD General Commands Manual          LS(1)

NAME
  ls -- list directory contents

SYNOPSIS
  ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1%] [file ...]

DESCRIPTION
  For each operand that names a file of a type other than directory,
  ls displays its name as well as any requested, associated informa-
  tion.  For each operand that names a file of type directory, ls dis-
  plays the names of files contained within that directory, as well as
  any requested, associated information.

  If no operands are given, the contents of the current directory are
  displayed.  If more than one operand is given, non-directory oper-
  ands are displayed first; directory and non-directory operands are
  sorted separately and in lexicographical order.

  The following options are available:

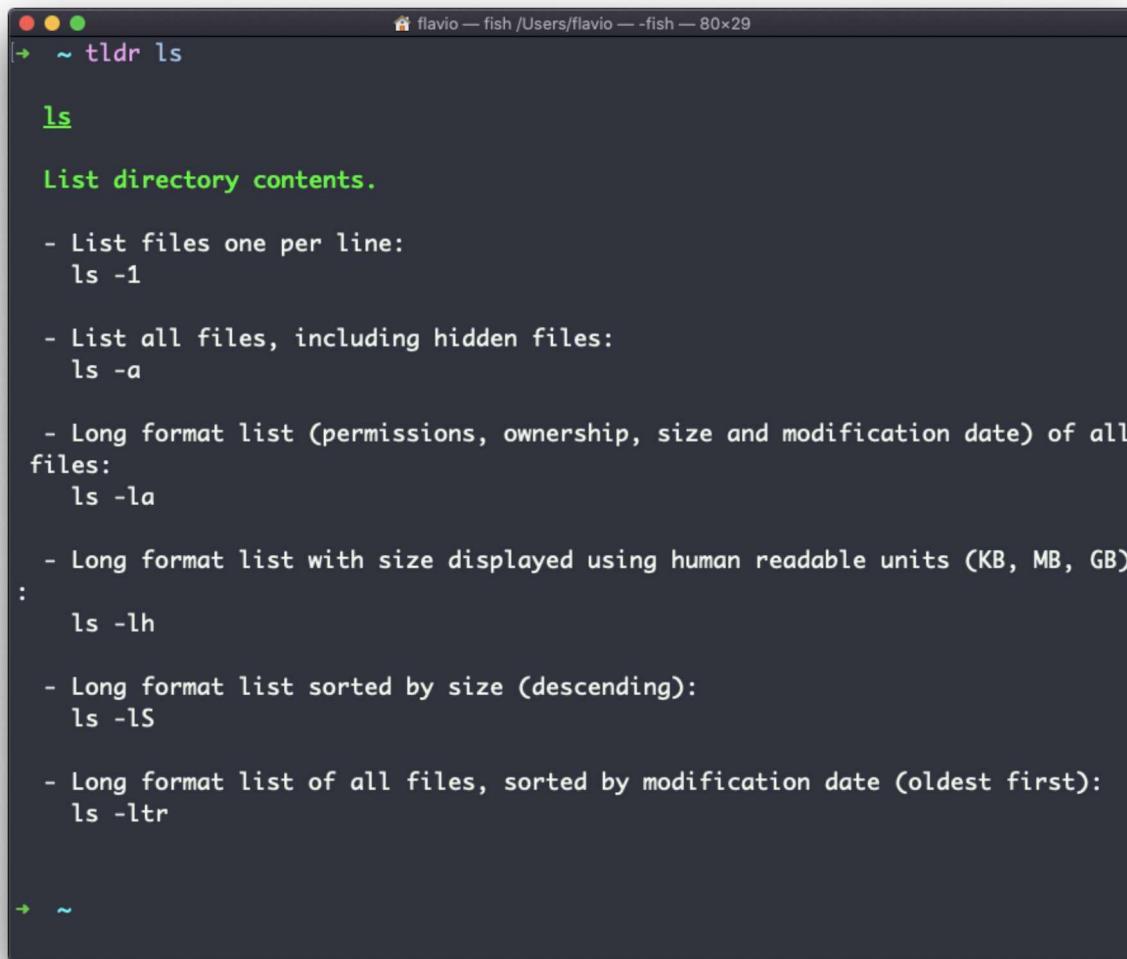
  -@      Display extended attribute keys and sizes in long (-l) out-
          put.

  -1      (The numeric digit ``one''.) Force output to be one entry
:
```

Esta é uma página de homem (do *manual*). As páginas de manual são uma ferramenta essencial para aprender, como desenvolvedor. Eles contêm tanta informação que às vezes é quase demais.

A captura de tela acima é apenas 1 de 14 telas de explicação para o comando.ls

Na maioria das vezes, quando estou precisando aprender um comando rapidamente, uso este site chamado **tldr pages**: <https://tldr.sh/>. É um comando que você pode instalar, então você o executa assim: , que fornece uma visão geral muito rápida de um comando, com alguns exemplos úteis de cenários de uso comuns:tldr <command>



A screenshot of a terminal window titled "flavio — fish /Users/flavio — -fish — 80x29". The command "tldr ls" is run, and the output is displayed. The output starts with the word "ls" in green, followed by a description: "List directory contents.". Below this, a list of options is provided, each with a command example:

- List files one per line:
ls -1
- List all files, including hidden files:
ls -a
- Long format list (permissions, ownership, size and modification date) of all files:
ls -la
- Long format list with size displayed using human readable units (KB, MB, GB)
ls -lh
- Long format list sorted by size (descending):
ls -lS
- Long format list of all files, sorted by modification date (oldest first):
ls -ltr

Este não é um substituto para , mas uma ferramenta útil para evitar perder-se na enorme quantidade de informações presentes em uma página de manual. Em seguida, você pode usar a página do manual para explorar todas as diferentes opções e parâmetros que você pode usar em um comando.man

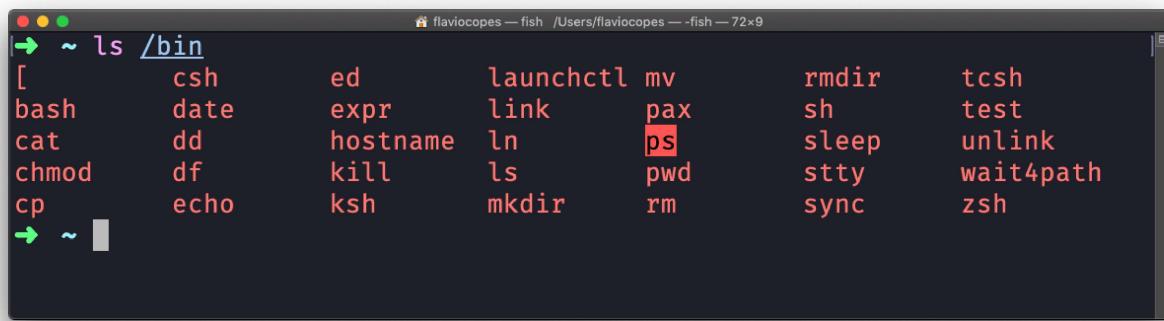
3. ls

Dentro de uma pasta, você pode listar todos os arquivos que a pasta contém usando o comando:ls

```
ls
```

Se você adicionar um nome ou caminho de pasta, ele imprimirá o conteúdo dessa pasta:

```
ls /bin
```



```
[ ~ ] ls /bin
[      csh      ed      launchctl mv      rmdir      tcsh
bash    date    expr    link      pax      sh      test
cat      dd    hostname  ln      ps      sleep      unlink
chmod   df    kill      ls      pwd      stty      wait4path
cp      echo    ksh      mkdir      rm      sync      zsh
[ ~ ]
```

ls aceita muitas opções. Uma das minhas combinações de opções favoritas é . Experimente:-al

```
ls -al /bin
```

```
~ ls -al /bin
total 5120
drwxr-xr-x@ 37 root wheel      1184 Feb  4 10:05 .
drwxr-xr-x  30 root wheel       960 Feb  8 15:32 ..
-rwxr-xr-x   1 root wheel     22704 Jan 16 02:21 [
-rxr-xr-xr-x  1 root wheel    618416 Jan 16 02:21 bash
-rwxr-xr-x   1 root wheel    23648 Jan 16 02:21 cat
-rwxr-xr-x   1 root wheel    34144 Jan 16 02:21 chmod
-rwxr-xr-x   1 root wheel    29024 Jan 16 02:21 cp
-rwxr-xr-x   1 root wheel   379952 Jan 16 02:21 csh
-rwxr-xr-x   1 root wheel   28608 Jan 16 02:21 date
-rwxr-xr-x   1 root wheel   32000 Jan 16 02:21 dd
-rwxr-xr-x   1 root wheel   23392 Jan 16 02:21 df
-rwxr-xr-x   1 root wheel   18128 Jan 16 02:21 echo
-rwxr-xr-x   1 root wheel   54080 Jan 16 02:21 ed
-rwxr-xr-x   1 root wheel   23104 Jan 16 02:21 expr
-rwxr-xr-x   1 root wheel   18288 Jan 16 02:21 hostname
-rwxr-xr-x   1 root wheel   18688 Jan 16 02:21 kill
-rxr-xr-xr-x  1 root wheel  1282864 Jan 16 02:21 ksh
-rwxr-xr-x   1 root wheel   121296 Jan 16 02:21 launchctl
```

em comparação com o simples, isso retorna muito mais informações.`ls`

Você tem, da esquerda para a direita:

- as permissões de arquivo (e se o sistema oferecer suporte a ACLs, você também receberá um sinalizador de ACL)
- o número de links para esse arquivo
- o proprietário do arquivo
- o grupo do arquivo
- o tamanho do arquivo em bytes
- o arquivo modificado datetime
- o nome do arquivo

Esse conjunto de dados é gerado pela opção. Em vez disso, a opção também mostra os arquivos ocultos.`la`

Arquivos ocultos são arquivos que começam com um ponto ()..

4. cd

Depois de ter uma pasta, você pode movê-la usando o comando. significa **c**hange **d**irectory. Você o invoca especificando uma pasta para a qual se

mover. Você pode especificar um nome de pasta ou um caminho inteiro.`cd cd`

Exemplo:

```
mkdir fruits  
cd fruits
```

Agora você está na pasta `fruits`

Você pode usar o caminho especial para indicar a pasta pai:..

```
cd .. #back to the home folder
```

O caractere `#` indica o início do comentário, que dura por toda a linha depois que ele é encontrado.

Você pode usá-lo para formar um caminho:

```
mkdir fruits  
mkdir cars  
cd fruits  
cd ../cars
```

Há outro indicador de caminho especial que é `,` e indica a pasta **atual**..

Você também pode usar caminhos absolutos, que começam a partir da pasta raiz:`/`

```
cd /etc
```

Este comando funciona em Linux, macOS, WSL e em qualquer lugar que você tenha um ambiente UNIX

5. `pwd`

Sempre que você se sentir perdido no sistema de arquivos, chame o comando para saber onde você está:`pwd`

```
pwd
```

Ele imprimirá o caminho da pasta atual.

6. `mkdir`

Criar pastas usando o comando:`mkdir`

```
mkdir fruits
```

Você pode criar várias pastas com um comando:

```
mkdir dogs cars
```

Você também pode criar várias pastas aninhadas adicionando a opção:`-p`

```
mkdir -p fruits/apples
```

As opções nos comandos UNIX geralmente assumem essa forma. Você os adiciona logo após o nome do comando e eles alteram a forma como o comando se comporta. Muitas vezes, você também pode combinar várias opções.

Você pode encontrar quais opções um comando oferece digitando . Tente agora com, por exemplo (pressione a tecla para esc a página do manual). As páginas de manual são a incrível ajuda interna para o UNIX.`man <commandname>`

7. `rmdir`

Assim como você pode criar uma pasta usando o `mkdir`, você pode excluir uma pasta usando: `rmdir`

```
mkdir fruits  
rmdir fruits
```

Você também pode excluir várias pastas de uma só vez:

```
mkdir fruits cars  
rmdir fruits cars
```

A pasta excluída deve estar vazia.

Para excluir pastas com arquivos nelas, usaremos o comando mais genérico que exclui arquivos e pastas, usando as opções: `rm -rf`

```
rm -rf fruits cars
```

Tenha cuidado, pois este comando não pede confirmação e removerá imediatamente qualquer coisa que você pedir para remover.

Não há **bin** ao remover arquivos da linha de comando, e recuperar arquivos perdidos pode ser difícil.

8. mv

Depois de ter um arquivo, você pode movê-lo usando o comando. Especifique o caminho atual do arquivo e seu novo caminho: `mv`

```
touch pear  
mv pear new_pear
```

O arquivo agora é movido para o `.` . É assim que você **renomeia** arquivos e pastas. `pear` `new_pear`

Se o último parâmetro for uma pasta, o arquivo localizado no caminho do primeiro parâmetro será movido para essa pasta. Nesse caso, você pode especificar uma lista de arquivos e todos eles serão movidos no caminho da pasta identificado pelo último parâmetro:

```
touch pear  
touch apple  
mkdir fruits  
mv pear apple fruits #pear and apple moved to the fruits folder
```

9. cp

Você pode copiar um arquivo usando o comando:cp

```
touch test  
cp apple another_apple
```

Para copiar pastas, você precisa adicionar a opção de copiar recursivamente todo o conteúdo da pasta:-r

```
mkdir fruits  
cp -r fruits cars
```

10. aberto

O comando permite que você abra um arquivo usando esta sintaxe:open

```
open <filename>
```

Você também pode abrir um diretório, que no macOS abre o aplicativo Finder com o diretório atual aberto:

```
open <directory name>
```

Eu uso o tempo todo para abrir o diretório atual:

```
open .
```

O símbolo especial aponta para o diretório atual, como aponta para o diretório pai...

O mesmo comando também pode ser usado para executar um aplicativo:

```
open <application name>
```

11. toque

Você pode criar um arquivo vazio usando o comando:`touch`

```
touch apple
```

Se o arquivo já existir, ele o abrirá no modo de gravação e o carimbo de data/hora do arquivo será atualizado.

12. encontrar

O comando pode ser usado para localizar arquivos ou pastas que correspondam a um padrão de pesquisa específico. Ele pesquisa recursivamente.`find`

Vamos aprender pelo exemplo.

Localize todos os arquivos na árvore atual que têm a extensão e imprima o caminho relativo de cada arquivo correspondente:`..js`

```
find . -name '*.js'
```

É importante usar aspas em torno de caracteres especiais para evitar que o shell os interprete.*

Encontre diretórios sob a árvore atual que correspondam ao nome "src":

```
find . -type d -name src
```

Use para pesquisar somente arquivos ou para pesquisar apenas links simbólicos.-type f-type l

-name diferencia maiúsculas de minúsculas. para executar uma pesquisa que não diferencia maiúsculas de minúsculas.-iname

Você pode pesquisar em várias árvores de raiz:

```
find folder1 folder2 -name filename.txt
```

Encontre diretórios sob a árvore atual que correspondam ao nome "node_modules" ou "público":

```
find . -type d -name node_modules -or -name public
```

Você também pode excluir um caminho, usando:-not -path

```
find . -type d -name '*.md' -not -path 'node_modules/*'
```

Você pode pesquisar arquivos que tenham mais de 100 caracteres (bytes) neles:

```
find . -type f -size +100c
```

Arquivos de pesquisa maiores que 100KB, mas menores que 1MB:

```
find . -type f -size +100k -size -1M
```

Arquivos de pesquisa editados há mais de 3 dias

```
find . -type f -mtime +3
```

Pesquisar arquivos editados nas últimas 24 horas

```
find . -type f -mtime -1
```

Você pode excluir todos os arquivos que correspondem a uma pesquisa adicionando a opção. Isso exclui todos os arquivos editados nas últimas 24 horas:-delete

```
find . -type f -mtime -1 -delete
```

Você pode executar um comando em cada resultado da pesquisa. Neste exemplo, executamos para imprimir o conteúdo do arquivo:cat

```
find . -type f -exec cat {} \;
```

observe o término . é preenchido com o nome do arquivo no momento da execução.\;{}

13. ln

O comando faz parte dos comandos do sistema de arquivos Linux.ln

Ele é usado para criar links. O que é um link? É como um ponteiro para outro arquivo. Um arquivo que aponta para outro arquivo. Você pode estar familiarizado com os atalhos do Windows. Eles são semelhantes.

Temos 2 tipos de links: **hard links** e **soft links**.

Links físicos raramente são usados. Eles têm algumas limitações: você não pode vincular a diretórios e não pode vincular a sistemas de arquivos externos (discos).

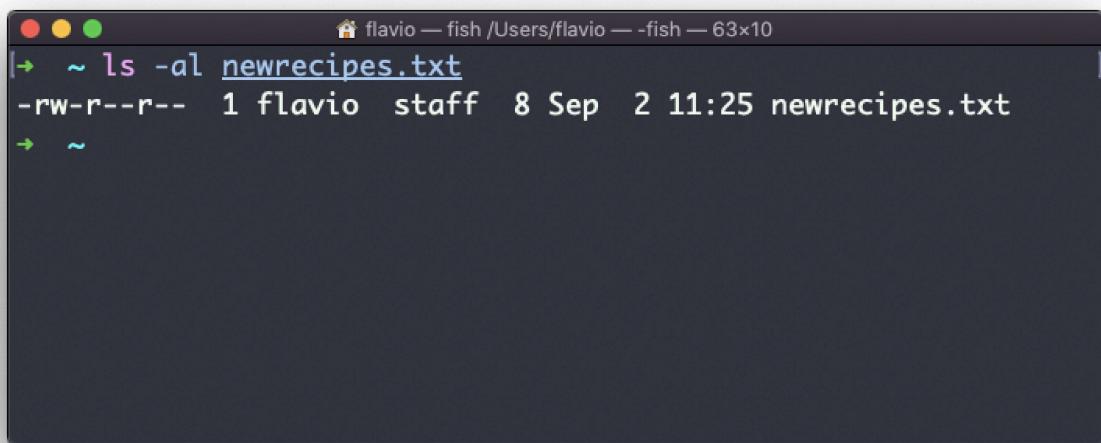
Um link físico é criado usando

```
ln <original> <link>
```

Por exemplo, digamos que você tenha um arquivo chamado `receitas.txt`. Você pode criar um link físico para ele usando:

```
ln recipes.txt newrecipes.txt
```

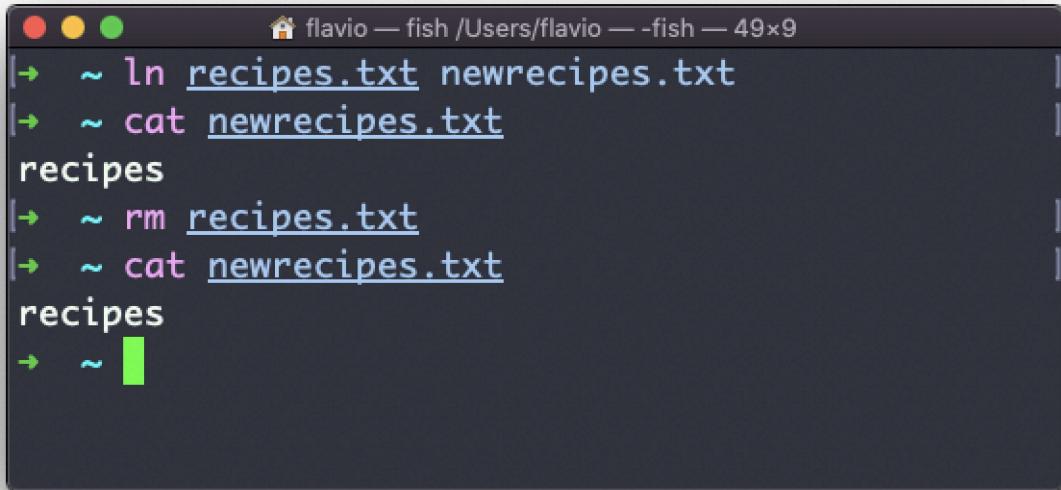
O novo link físico que você criou é indistinguível de um arquivo normal:



```
flavio — fish /Users/flavio — -fish — 63x10
ls -al newrecipes.txt
-rw-r--r--  1 flavio  staff  8 Sep  2 11:25 newrecipes.txt
~
```

Agora, sempre que você editar qualquer um desses arquivos, o conteúdo será atualizado para ambos.

Se você excluir o arquivo original, o link ainda conterá o conteúdo do arquivo original, pois isso não será removido até que haja um link físico apontando para ele.



```
flavio — fish /Users/flavio — -fish — 49x9
|~ ln recipes.txt newrecipes.txt
|~ cat newrecipes.txt
recipes
|~ rm recipes.txt
|~ cat newrecipes.txt
recipes
~
```

Soft links são diferentes. Eles são mais poderosos, pois você pode vincular a outros sistemas de arquivos e diretórios, mas quando o original for removido, o link será quebrado.

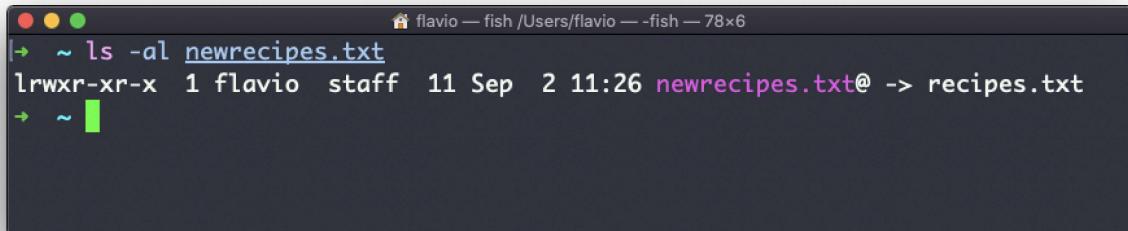
Você cria soft links usando a opção de:`-sln`

```
ln -s <original> <link>
```

Por exemplo, digamos que você tenha um arquivo chamado `receitas.txt`. Você pode criar um link flexível para ele usando:

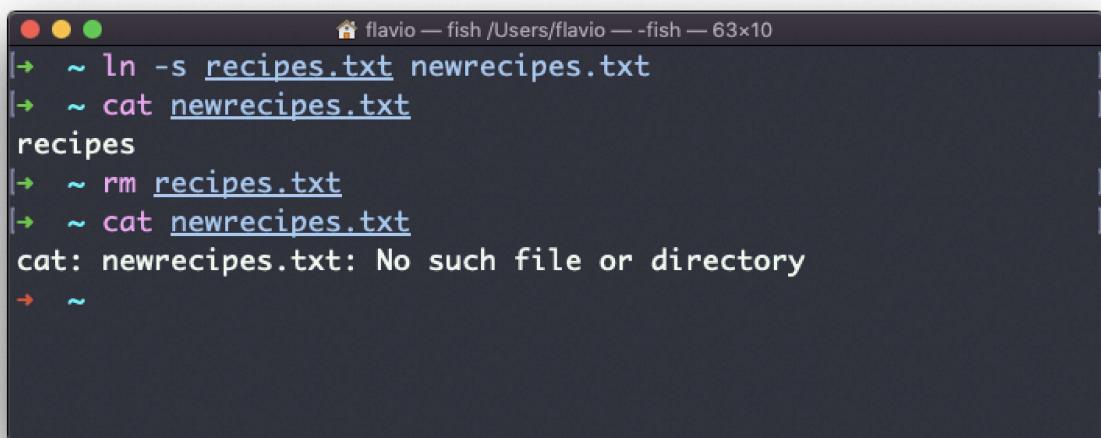
```
ln -s recipes.txt newrecipes.txt
```

Nesse caso, você pode ver que há um sinalizador especial quando você lista o arquivo usando o `ls`, e o nome do arquivo tem um `@` no final, e é colorido de forma diferente se você tiver as cores habilitadas:`ls -al@`



```
flavio — fish /Users/flavio — -fish — 78x6
ls -al newrecipes.txt
lrwxr-xr-x 1 flavio staff 11 Sep 2 11:26 newrecipes.txt@ -> recipes.txt
~
```

Agora, se você excluir o arquivo original, os links serão quebrados e o shell informará "Não há tal arquivo ou diretório" se você tentar acessá-lo:



```
ln -s recipes.txt newrecipes.txt
cat newrecipes.txt
recipes
rm recipes.txt
cat newrecipes.txt
cat: newrecipes.txt: No such file or directory
~
```

14. gzip

Você pode compactar um arquivo usando o protocolo de compactação gzip chamado LZ77 usando o comando `gzip`

Aqui está o uso mais simples:

```
gzip filename
```

Isso compactará o arquivo e acrescentará uma extensão a ele. O arquivo original é excluído. Para evitar isso, você pode usar a opção `e` e usar o redirecionamento de saída para gravar a saída no arquivo: `.gz -c filename.gz`

```
gzip -c filename > filename.gz
```

A opção especifica que a saída irá para o fluxo de saída padrão, deixando o arquivo original intacto -c

Ou você pode usar a opção:-k

```
gzip -k filename
```

Existem vários níveis de compressão. Quanto mais a compressão, mais tempo levará para comprimir (e descomprimir). Os níveis variam de 1 (mais rápido, pior compressão) a 9 (mais lento, melhor compressão), e o padrão é 6.

Você pode escolher um nível específico com a opção:-<NUMBER>

```
gzip -1 filename
```

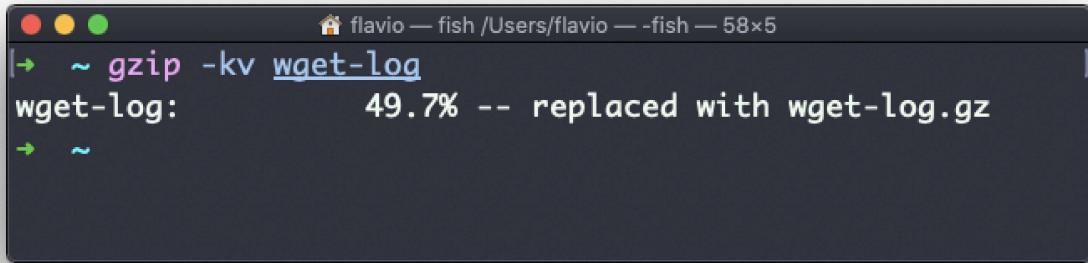
Você pode compactar vários arquivos listando-os:

```
gzip filename1 filename2
```

Você pode compactar todos os arquivos em um diretório, recursivamente, usando a opção:-r

```
gzip -r a_folder
```

A opção imprime as informações de porcentagem de compactação. Aqui está um exemplo de como ele está sendo usado junto com a opção (manter):-v-k



```
flavio — fish /Users/flavio — -fish — 58x5
|~ gzip -kv wget-log
wget-log:          49.7% -- replaced with wget-log.gz
|~
```

gzip também pode ser usado para descompactar um arquivo, usando a opção:-d

```
gzip -d filename.gz
```

15. Gunzip

O comando é basicamente equivalente ao comando, exceto que a opção está sempre habilitada por padrão.`gunzip``gzip -d`

O comando pode ser invocado da seguinte maneira:

```
gunzip filename.gz
```

Isso irá gunzip e irá remover a extensão, colocando o resultado no arquivo. Se esse arquivo existir, ele substituirá isso..`gzfilename`

Você pode extrair para um nome de arquivo diferente usando o redirecionamento de saída usando a opção:`-c`

```
gunzip -c filename.gz > anotherfilename
```

16.tar

O comando é usado para criar um arquivo, agrupando vários arquivos em um único arquivo.`.tar`

Seu nome vem do passado e significa *arquivo de fita*. Quando os arquivos eram armazenados em fitas.

Este comando cria um arquivo morto nomeado com o conteúdo de e :archive.tarfile1file2

```
tar -cf archive.tar file1 file2
```

A opção significa *criar*. A opção é usada para gravar no arquivo morto.cf

Para extrair arquivos de um arquivo morto na pasta atual, use:

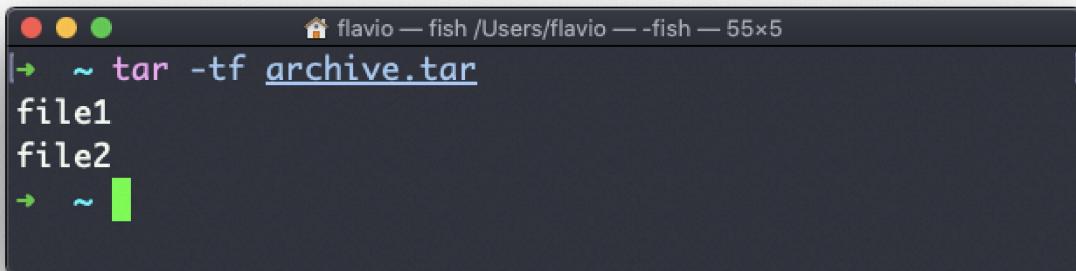
```
tar -xf archive.tar
```

a opção significa *extratox*

e para extraí-los para um diretório específico, use:

```
tar -xf archive.tar -C directory
```

Você também pode apenas listar os arquivos contidos em um arquivo:



```
flavio — fish /Users/flavio — -fish — 55×5
|~ tar -tf archive.tar
file1
file2
~ |
```

tar é frequentemente usado para criar um arquivo **compactado**, gzipping o arquivo.

Isso é feito usando a opção:z

```
tar -czf archive.tar.gz file1 file2
```

Isso é como criar um arquivo tar e, em seguida, executar nele.gzip

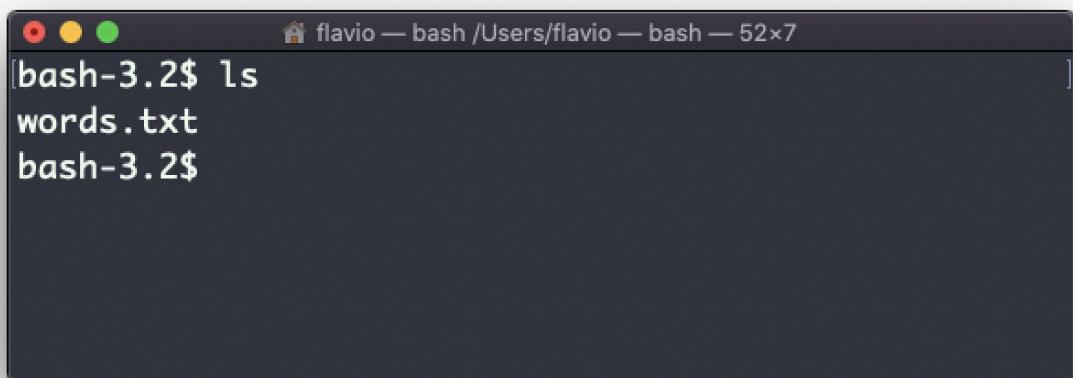
Para desarquivar um arquivo compactado, você pode usar o ou , ou , e depois desarquivá-lo, mas reconhecerá que é um arquivo compactado e fará isso por você:
gunzipgzip -dtar -xf

```
tar -xf archive.tar.gz
```

17. pseudônimo

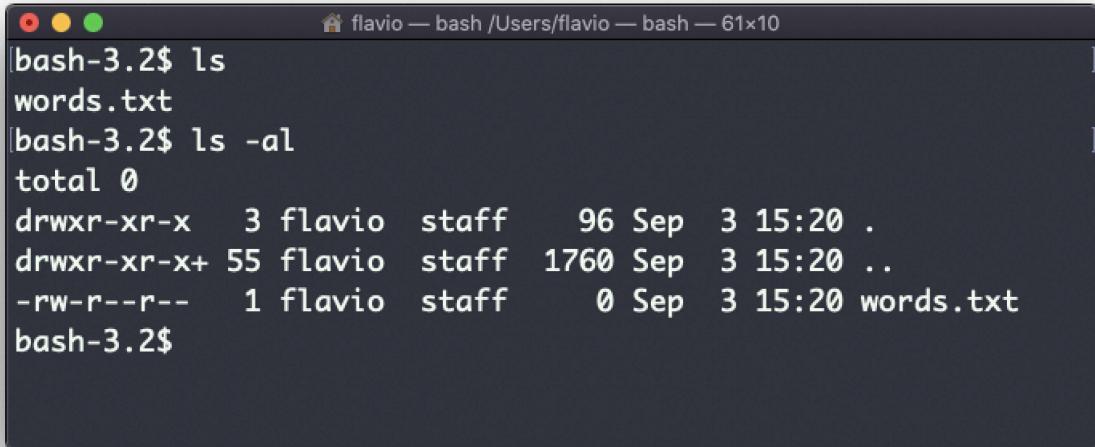
É comum sempre executar um programa com um conjunto de opções que você gosta de usar.

Por exemplo, pegue o comando. Por padrão, ele imprime muito pouca informação:
ls



```
flavio — bash /Users/flavio — bash — 52x7
[bash-3.2$ ls
words.txt
bash-3.2$
```

ao usar a opção, ele imprimirá algo mais útil, incluindo a data de modificação do arquivo, o tamanho, o proprietário e as permissões, listando também arquivos ocultos (arquivos que começam com um :).
-al



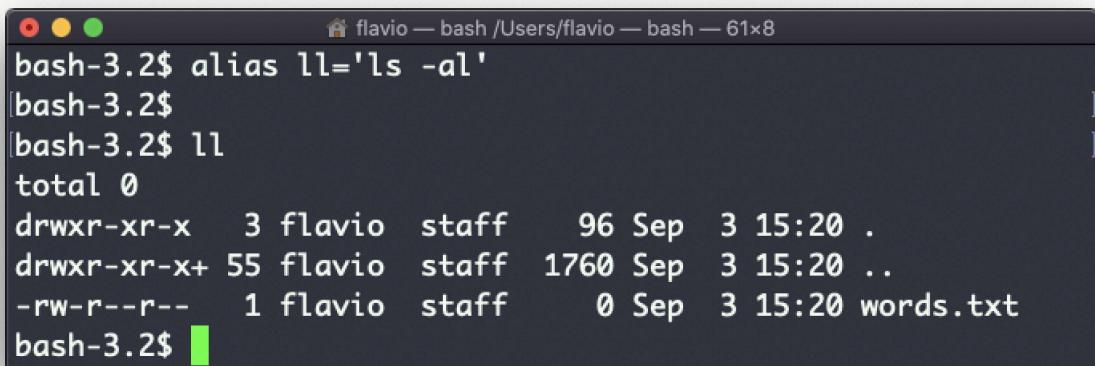
```
fladio — bash /Users/fladio — bash — 61x10
[bash-3.2$ ls
words.txt
[bash-3.2$ ls -al
total 0
drwxr-xr-x  3 flavio  staff   96 Sep  3 15:20 .
drwxr-xr-x+ 55 flavio  staff  1760 Sep  3 15:20 ..
-rw-r--r--  1 flavio  staff     0 Sep  3 15:20 words.txt
bash-3.2$
```

Você pode criar um novo comando, por exemplo, eu gosto de chamá-lo de , que é um alias para .llls -al

Você faz isso desta maneira:

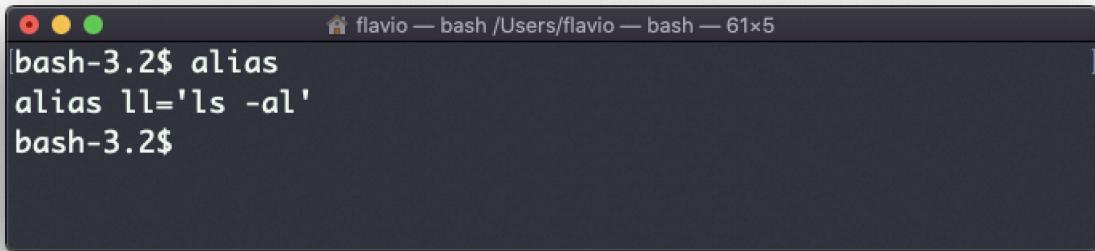
```
alias ll='ls -al'
```

Depois de fazer isso, você pode chamar como se fosse um comando UNIX regular:ll



```
fladio — bash /Users/fladio — bash — 61x8
[bash-3.2$ alias ll='ls -al'
[bash-3.2$ ll
total 0
drwxr-xr-x  3 flavio  staff   96 Sep  3 15:20 .
drwxr-xr-x+ 55 flavio  staff  1760 Sep  3 15:20 ..
-rw-r--r--  1 flavio  staff     0 Sep  3 15:20 words.txt
bash-3.2$
```

Now calling without any option will list the aliases defined:alias



```
fladio — bash /Users/fladio — bash — 61x5
bash-3.2$ alias
alias ll='ls -al'
bash-3.2$
```

O alias funcionará até que a sessão do terminal seja fechada.

Para torná-lo permanente, você precisa adicioná-lo à configuração do shell, que pode ser ou se você usar o shell Bash, dependendo do caso de uso.`~/.bashrc``~/.profile``~/.bash_profile`

Tenha cuidado com aspas se você tiver variáveis no comando: usando aspas duplas a variável é resolvida no momento da definição, usando aspas simples é resolvida no momento da invocação. Esses 2 são diferentes:

```
alias lsthis="ls $PWD"
alias lscurrent='ls $PWD'
```

`$PWD` refere-se à pasta atual em que o shell está. Se agora você navegar para uma nova pasta, lista os arquivos na nova pasta, ainda lista os arquivos na pasta que você estava quando definiu o alias.`lscurrent``lsthis`

18. gato

Semelhante à cauda de alguma forma, temos . Exceto também pode adicionar conteúdo a um arquivo, e isso o torna super poderoso.`cat``cat`

Em seu uso mais simples, imprime o conteúdo de um arquivo na saída padrão:`cat`

```
cat file
```

Você pode imprimir o conteúdo de vários arquivos:

```
cat file1 file2
```

e usando o operador de redirecionamento de saída, você pode concatenar o conteúdo de vários arquivos em um novo arquivo:>

```
cat file1 file2 > file3
```

Usando você pode acrescentar o conteúdo de vários arquivos em um novo arquivo, criando-o se ele não existir:>>

```
cat file1 file2 >> file3
```

Ao assistir a arquivos de código-fonte, é ótimo ver os números de linha, e você pode imprimi-los usando a opção:`cat -n`

```
cat -n file1
```

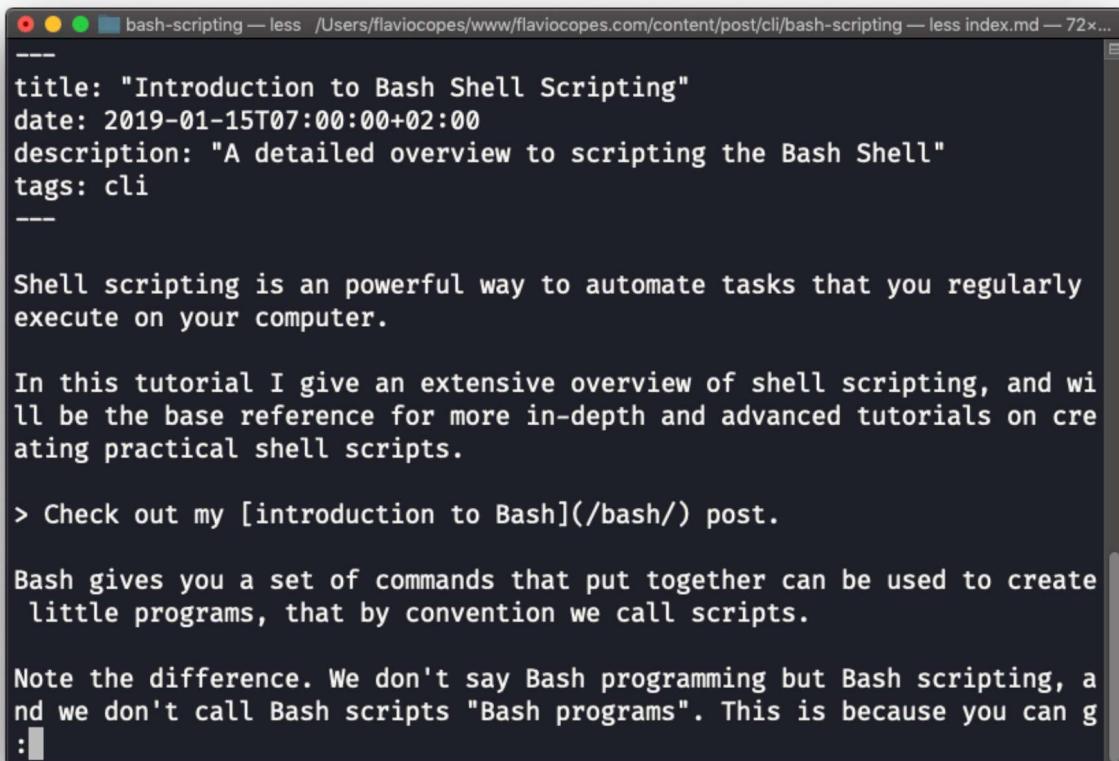
Você só pode adicionar um número a linhas que não estejam em branco usando o `-n`, ou também pode remover todas as várias linhas vazias usando o `-b -s`.

`cat` é frequentemente usado em combinação com o operador de pipe para alimentar o conteúdo de um arquivo como entrada para outro comando:
`.|cat file1 | anothercommand`

19. menos

O comando é um que eu uso muito. Ele mostra o conteúdo armazenado dentro de um arquivo, em uma interface de usuário agradável e interativa.`less`

Uso:`:less <filename>`



A screenshot of a terminal window titled "bash-scripting — less". The window displays the contents of a file named "index.md". The text includes metadata at the top:

```
title: "Introduction to Bash Shell Scripting"
date: 2019-01-15T07:00:00+02:00
description: "A detailed overview to scripting the Bash Shell"
tags: cli
```

Followed by a section about shell scripting:

```
Shell scripting is an powerful way to automate tasks that you regularly execute on your computer.
```

Text explaining the purpose of shell scripts:

```
In this tutorial I give an extensive overview of shell scripting, and will be the base reference for more in-depth and advanced tutorials on creating practical shell scripts.
```

Text about Bash scripts:

```
> Check out my [introduction to Bash](/bash/) post.
```

Text about Bash programs:

```
Bash gives you a set of commands that put together can be used to create little programs, that by convention we call scripts.
```

Text about Bash scripting vs Bash programming:

```
Note the difference. We don't say Bash programming but Bash scripting, and we don't call Bash scripts "Bash programs". This is because you can g :|
```

Uma vez que você está dentro de uma sessão, você pode sair pressionando `.lessq`

Você pode navegar pelo conteúdo do arquivo usando as teclas `e` ou `o` para navegar página por página. Você também pode pular para o final do arquivo pressionando `e` e voltar para o início pressionando `.updownspace` barba `Gg`

Você pode pesquisar o conteúdo dentro do arquivo pressionando `e` digitando uma palavra para pesquisar. Isso pesquisa *para frente*. Você pode pesquisar para trás usando o símbolo `e` e digitando uma palavra `./?`

Este comando apenas visualiza o conteúdo do arquivo. Você pode abrir diretamente um editor pressionando `.v`. Ele usará o editor do sistema, que na maioria dos casos é `.vim`

Pressionar a tecla `v` entra no modo de acompanhamento ou *no modo de exibição*. Quando o arquivo é alterado por outra pessoa, como de outro programa, você pode ver as alterações *ao vivo*. Por padrão, isso não está acontecendo e você só vê a versão do arquivo no momento em que o abriu.

Você precisa pressionar para sair deste modo. Nesse caso, o comportamento é semelhante ao de executar o comando.`Fctrl-Ctail -f <filename>`

Você pode abrir vários arquivos e navegar por eles usando (para ir para o próximo arquivo) e (para ir para o anterior).`:n:p`

20. cauda

O melhor caso de uso de cauda na minha opinião é quando chamado com a opção. Ele abre o arquivo no final e observa as alterações do arquivo. Sempre que houver novo conteúdo no arquivo, ele será impresso na janela. Isso é ótimo para observar arquivos de log, por exemplo:`-f`

```
tail -f /var/log/system.log
```

Para sair, pressione `.ctrl-C`

Você pode imprimir as últimas 10 linhas em um arquivo:

```
tail -n 10 <filename>
```

Você pode imprimir todo o conteúdo do arquivo a partir de uma linha específica usando antes do número da linha:`+`

```
tail -n +10 <filename>
```

`tail` pode fazer muito mais e como sempre o meu conselho é verificar.`man tail`

21. wc

O comando nos dá informações úteis sobre um arquivo ou entrada que ele recebe via pipes.`wc`

```
echo test >> test.txt  
wc test.txt  
1 1 5 test.txt
```

Exemplo via pipes, podemos contar a saída da execução do comando:ls -al

```
ls -al | wc  
6 47 284
```

A primeira coluna retornada é o número de linhas. O segundo é o número de palavras. O terceiro é o número de bytes.

Podemos dizer-lhe para apenas contar as linhas:

```
wc -l test.txt
```

ou apenas as palavras:

```
wc -w test.txt
```

ou apenas os bytes:

```
wc -c test.txt
```

Bytes em conjuntos de caracteres ASCII equivalem a caracteres, mas com conjuntos de caracteres não-ASCII, o número de caracteres pode diferir porque alguns caracteres podem levar vários bytes, por exemplo, isso acontece em Unicode.

Nesse caso, o sinalizador ajudará a obter o valor correto:-m

```
wc -m test.txt
```

22. grep

O comando é uma ferramenta muito útil, que quando você dominar irá ajudá-lo tremendamente no seu dia a dia.grep

Se você está se perguntando, significa *impressão de expressão regular global* grep

Você pode usar para pesquisar em arquivos ou combiná-lo com pipes para filtrar a saída de outro comando.grep

Por exemplo, aqui está como podemos encontrar as ocorrências da linha no arquivo:document.getElementById index.md

```
grep document.getElementById index.md
```



```
flavio — bash /Users/flavio — bash — 77x7
bash-3.2$ grep document.getElementById index.md
document.getElementById('button').addEventListener('click', () => {
  document.getElementById('button').addEventListener('click', () => {
bash-3.2$
```

Usando a opção, ele mostrará os números das linhas:-n

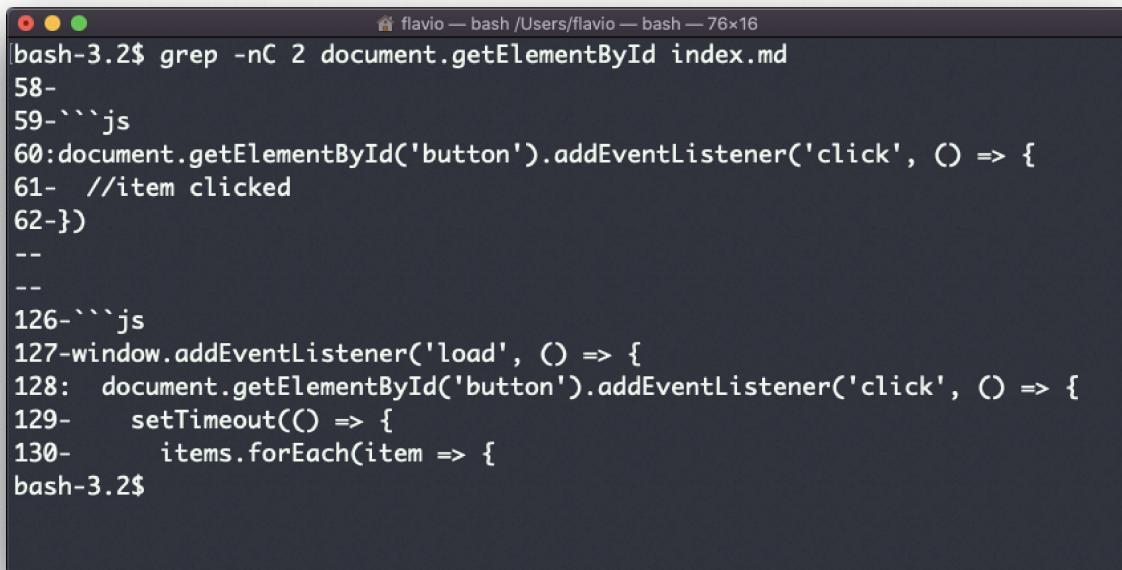
```
grep -n document.getElementById index.md
```



```
flavio — bash /Users/flavio — bash — 77x7
bash-3.2$ grep -n document.getElementById index.md
60:document.getElementById('button').addEventListener('click', () => {
128:  document.getElementById('button').addEventListener('click', () => {
bash-3.2$
```

Uma coisa muito útil é dizer grep para imprimir 2 linhas antes, e 2 linhas depois da linha correspondente, para nos dar mais contexto. Isso é feito usando a opção, que aceita várias linhas:-C

```
grep -nC 2 document.getElementById index.md
```

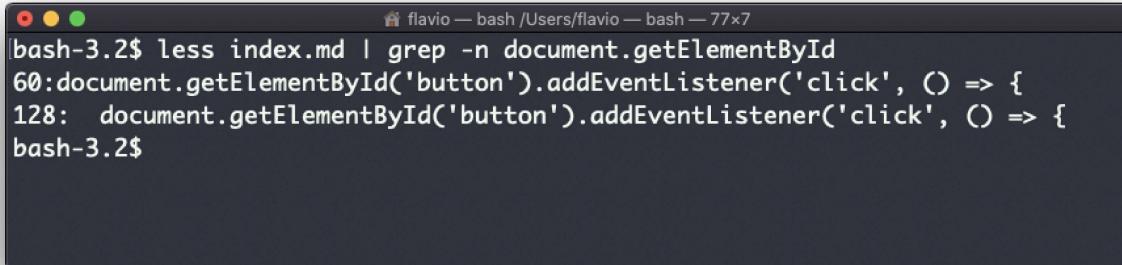


```
flavio — bash /Users/flavio — bash — 76x16
bash-3.2$ grep -nC 2 document.getElementById index.md
58-
59-```js
60:document.getElementById('button').addEventListener('click', () => {
61-   //item clicked
62-})
--
--
126-```js
127>window.addEventListener('load', () => {
128:   document.getElementById('button').addEventListener('click', () => {
129-     setTimeout(() => {
130-       items.forEach(item => {
bash-3.2$
```

A pesquisa diferencia maiúsculas de minúsculas por padrão. Use o sinalizador para torná-lo insensível.-i

Como mencionado, você pode usar grep para filtrar a saída de outro comando. Podemos replicar a mesma funcionalidade acima usando:

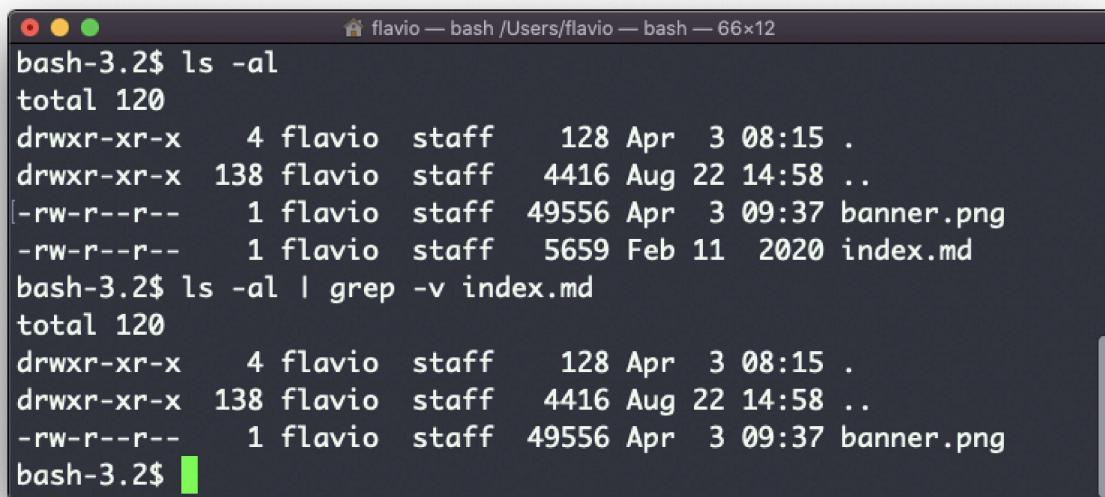
```
less index.md | grep -n document.getElementById
```



```
flavio — bash /Users/flavio — bash — 77x7
bash-3.2$ less index.md | grep -n document.getElementById
60:document.getElementById('button').addEventListener('click', () => {
128:   document.getElementById('button').addEventListener('click', () => {
bash-3.2$
```

A cadeia de caracteres de pesquisa pode ser uma expressão regular, e isso torna muito poderoso.grep

Outra coisa que você pode achar muito útil é inverter o resultado, excluindo as linhas que correspondem a uma cadeia de caracteres específica, usando a opção:-v



The screenshot shows a terminal window titled "flavio — bash /Users/flavio — bash — 66x12". It displays two commands: "ls -al" and "ls -al | grep -v index.md". The first command shows a directory listing with three files: a dot file, a double-dot file, and a file named "banner.png". The second command filters out the "index.md" file from the listing, leaving only the other two files.

```
bash-3.2$ ls -al
total 120
drwxr-xr-x    4 flavio  staff      128 Apr  3 08:15 .
drwxr-xr-x  138 flavio  staff     4416 Aug 22 14:58 ..
[-rw-r--r--    1 flavio  staff   49556 Apr  3 09:37 banner.png
-rw-r--r--    1 flavio  staff    5659 Feb 11  2020 index.md
bash-3.2$ ls -al | grep -v index.md
total 120
drwxr-xr-x    4 flavio  staff      128 Apr  3 08:15 .
drwxr-xr-x  138 flavio  staff     4416 Aug 22 14:58 ..
[-rw-r--r--    1 flavio  staff   49556 Apr  3 09:37 banner.png
bash-3.2$
```

23. classificar

Suponha que você tenha um arquivo de texto que contenha os nomes dos cães:

```
Roger
Syd
Vanille
Luna
Ivica
Tina
~  
~  
~  
~  
(END)
```

Esta lista não está ordenada.

O comando nos ajuda a classificá-los por nome:`sort`

```
|~ sort dogs.txt
Ivica
Luna
Roger
Syd
Tina
Vanille
→ ~ |
```

Use a opção para inverter a ordem:`r`

```
flavio — fish /Users/flavio — -fish — 51x9
|~ sort -r dogs.txt
Vanille
Tina
Syd
Roger
Luna
Ivica
→ ~
```

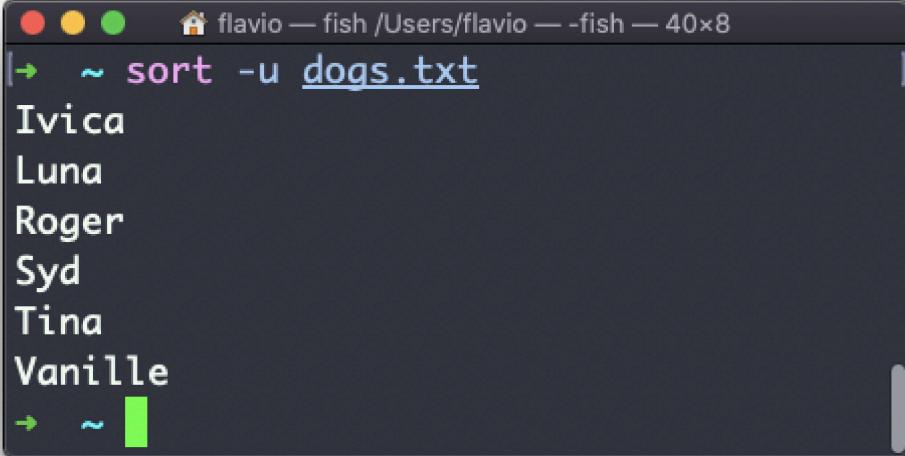
A classificação por padrão diferencia maiúsculas de minúsculas e é alfabética.
Use a opção para classificar maiúsculas e minúsculas e a opção para
classificar usando uma ordem numérica.`--ignore-case-n`

Se o arquivo contiver linhas duplicadas:

```
flavio — nano /Users/flavio — nano dogs.txt — 60x15
GNU nano 2.0.6          File: dogs.txt          Modified
Roger
Syd
Vanille
Luna
Ivica
Tina
Roger
Syd

[ Cancelled ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit      ^J Justify ^W Where ^V Next Page ^U Uncut T^T To Spell
```

Você pode usar a opção para removê-los:`-u`



```
flavio — fish /Users/flavio — -fish — 40x8
|~ sort -u dogs.txt
Ivica
Luna
Roger
Syd
Tina
Vanille
→ ~
```

sort não funciona apenas em arquivos, como muitos comandos UNIX ele também funciona com pipes, para que você possa usar na saída de outro comando, por exemplo, você pode ordenar os arquivos retornados por:ls

```
ls | sort
```

sort é muito poderoso e tem muito mais opções, que você pode explorar chamando `.man sort`

The screenshot shows a terminal window with the title "flavio — man /Users/flavio — less - man sort — 115x55". The content is the man page for the sort command, which is part of the BSD General Commands Manual. The page is titled "SORT(1)" and covers topics such as NAME, SYNOPSIS, DESCRIPTION, and various command-line options like -c, -m, -o, -S, -T, -u, -s, --version, and :.

```

      SORT(1)          BSD General Commands Manual          SORT(1)

NAME
   sort -- sort or merge records (lines) of text and binary files

SYNOPSIS
   sort [-bcCdfghiRMmnrsuVz] [-k field1[,field2]] [-S memsize] [-T dir] [-t char] [-o output]
         [file ...]
   sort --help
   sort --version

DESCRIPTION
   The sort utility sorts text and binary files by lines. A line is a record separated from the subsequent record by a newline (default) or NUL '\0' character (-z option). A record can contain any printable or unprintable characters. Comparisons are based on one or more sort keys extracted from each line of input, and are performed lexicographically, according to the current locale's collating rules and the specified command-line options that can tune the actual sorting behavior. By default, if keys are not given, sort uses entire lines for comparison.

   The command line options are as follows:

   -c, --check, -C, --check=silent|quiet
      Check that the single input file is sorted. If the file is not sorted, sort produces the appropriate error messages and exits with code 1, otherwise returns 0. If -C or --check=silent is specified, sort produces no output. This is a "silent" version of -c.

   -m, --merge
      Merge only. The input files are assumed to be pre-sorted. If they are not sorted the output order is undefined.

   -o output, --output=output
      Print the output to the output file instead of the standard output.

   -S size, --buffer-size=size
      Use size for the maximum size of the memory buffer. Size modifiers %,b,K,M,G,T,P,E,Z,Y can be used. If a memory limit is not explicitly specified, sort takes up to about 90% of available memory. If the file size is too big to fit into the memory buffer, the temporary disk files are used to perform the sorting.

   -T dir, --temporary-directory=dir
      Store temporary files in the directory dir. The default path is the value of the environment variable TMPDIR or /var/tmp if TMPDIR is not defined.

   -u, --unique
      Unique keys. Suppress all lines that have a key that is equal to an already processed one. This option, similarly to -s, implies a stable sort. If used with -c or -C, sort also checks that there are no lines with duplicate keys.

   -s
      Stable sort. This option maintains the original record order of records that have an equal key. This is a non-standard feature, but it is widely accepted and used.

   --version
      Print the version and silently exits.

:

```

24. uniq

uniq é um comando útil para classificar linhas de texto.

Você pode obter essas linhas de um arquivo ou usando pipes da saída de outro comando:

```
uniq dogs.txt
```

```
ls | uniq
```

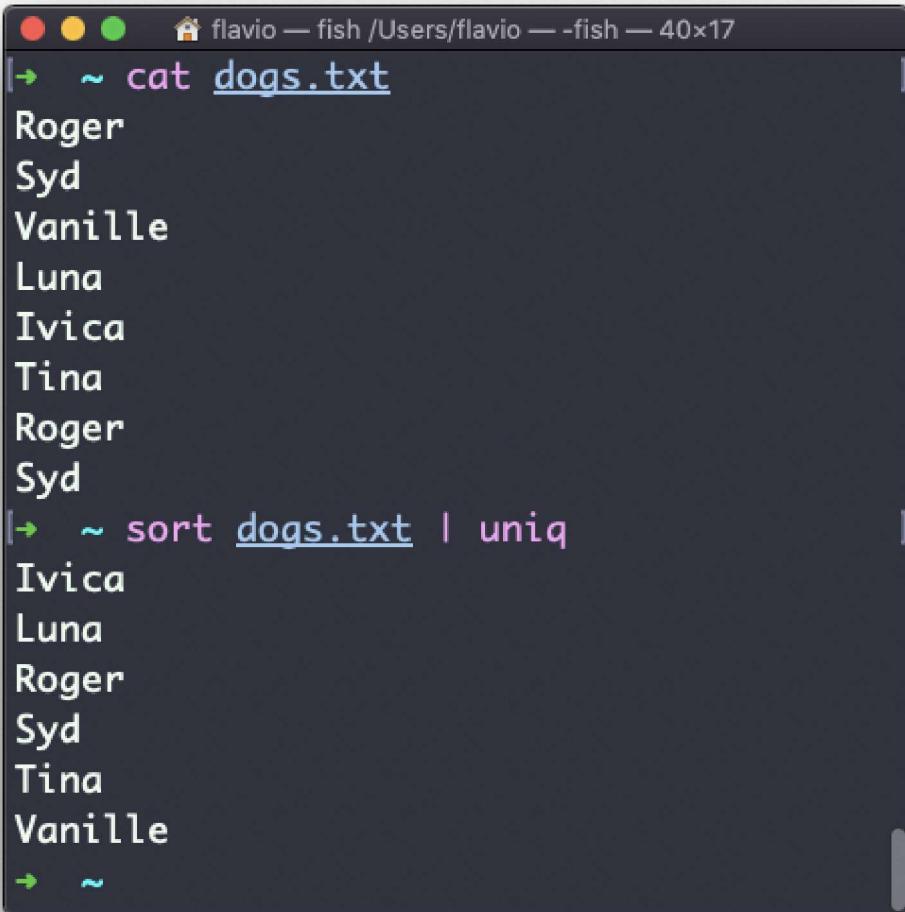
Você precisa considerar esta coisa chave: só detectará linhas duplicadas adjacentes.`uniq`

Isso implica que você provavelmente o usará junto com:`:sort`

```
sort dogs.txt | uniq
```

O comando tem sua própria maneira de remover duplicatas com a opção (*exclusiva*). Mas tem mais poder.`sort -u``uniq`

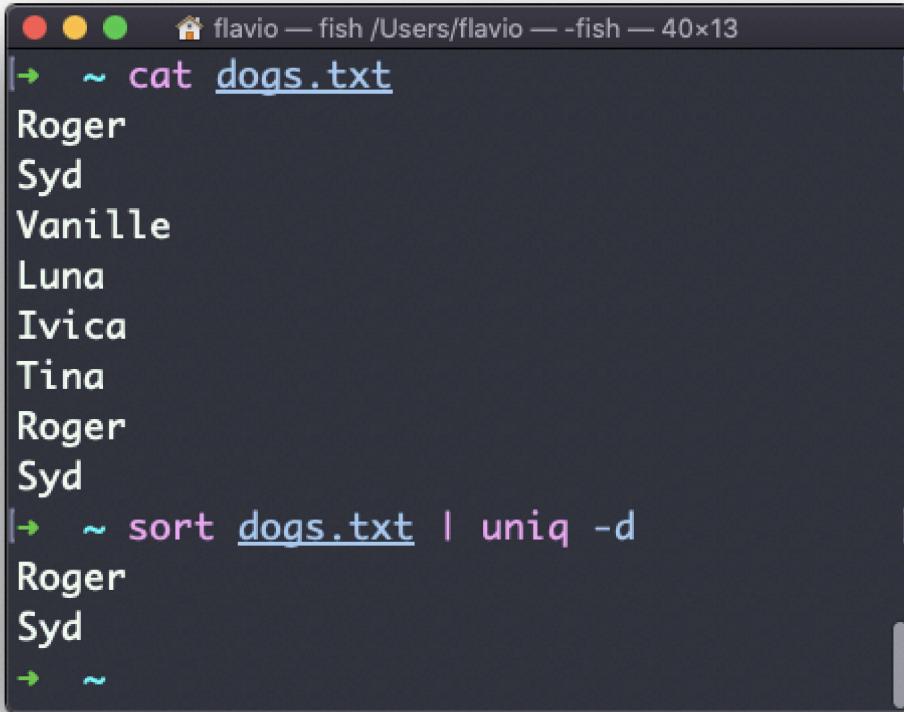
Por padrão, ele remove linhas duplicadas:



```
flavio — fish /Users/flavio — -fish — 40x17
|~ cat dogs.txt
Roger
Syd
Vanille
Luna
Ivica
Tina
Roger
Syd
|~ sort dogs.txt | uniq
Ivica
Luna
Roger
Syd
Tina
Vanille
|~
```

Você pode dizer a ele para exibir apenas linhas duplicadas, por exemplo, com a opção:`-d`

```
sort dogs.txt | uniq -d
```



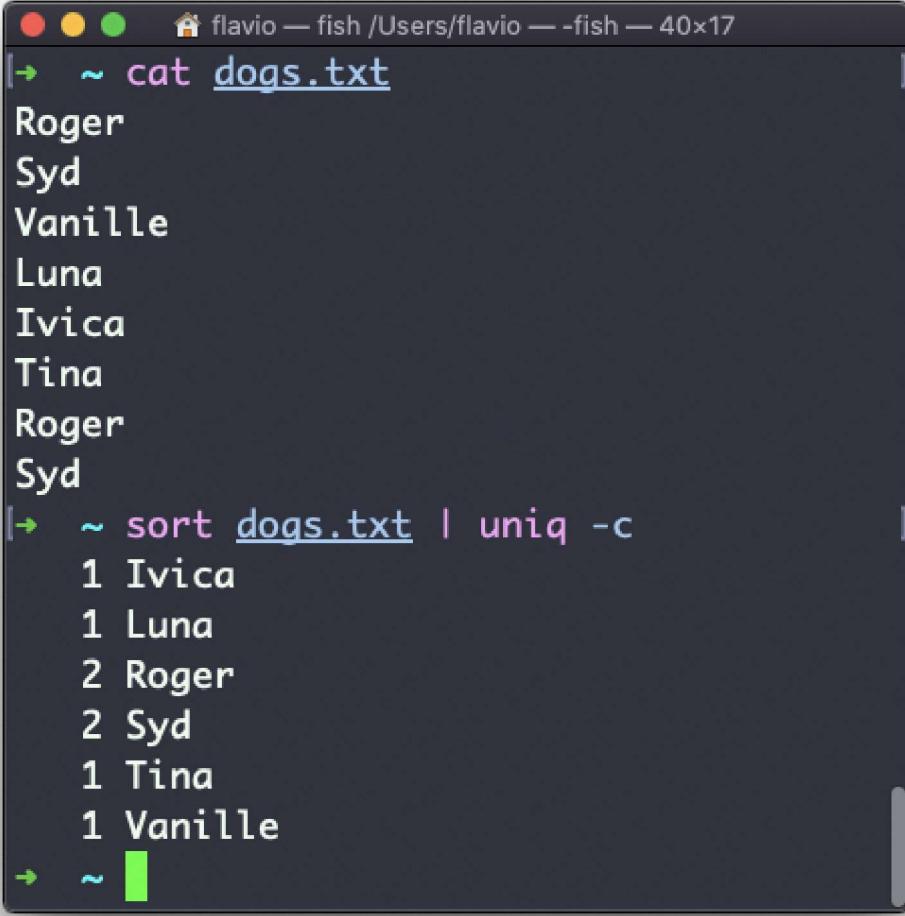
The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "flavio — fish /Users/flavio — -fish — 40x13". The user runs the command "cat dogs.txt", which outputs the following names: Roger, Syd, Vanille, Luna, Ivica, Tina, Roger, Syd. Then, the user runs "sort dogs.txt | uniq -d", which outputs the same unique names: Roger, Syd.

```
|~ cat dogs.txt
Roger
Syd
Vanille
Luna
Ivica
Tina
Roger
Syd
|~ sort dogs.txt | uniq -d
Roger
Syd
|~
```

Você pode usar a opção para exibir apenas linhas não duplicadas:-u

```
flavio — fish /Users/flavio — -fish — 40x15
|~ cat dogs.txt
Roger
Syd
Vanille
Luna
Ivica
Tina
Roger
Syd
|~ sort dogs.txt | uniq -u
Ivica
Luna
Tina
Vanille
→ ~
```

Você pode contar as ocorrências de cada linha com a opção:-c



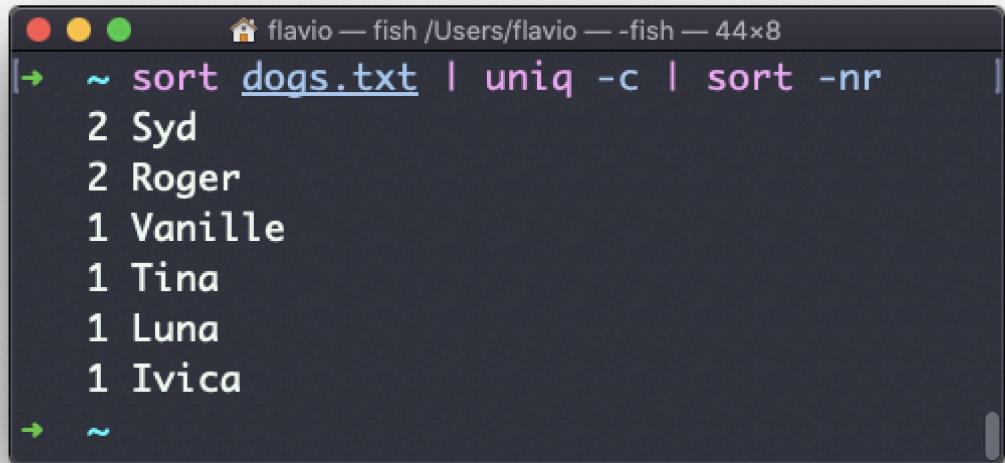
The screenshot shows a terminal window with the following session:

```
flavio — fish /Users/flavio — -fish — 40x17
~ cat dogs.txt
Roger
Syd
Vanille
Luna
Ivica
Tina
Roger
Syd
~ sort dogs.txt | uniq -c
1 Ivica
1 Luna
2 Roger
2 Syd
1 Tina
1 Vanille
~
```

Use a combinação especial:

```
sort dogs.txt | uniq -c | sort -nr
```

para então classificar essas linhas por mais frequentes:



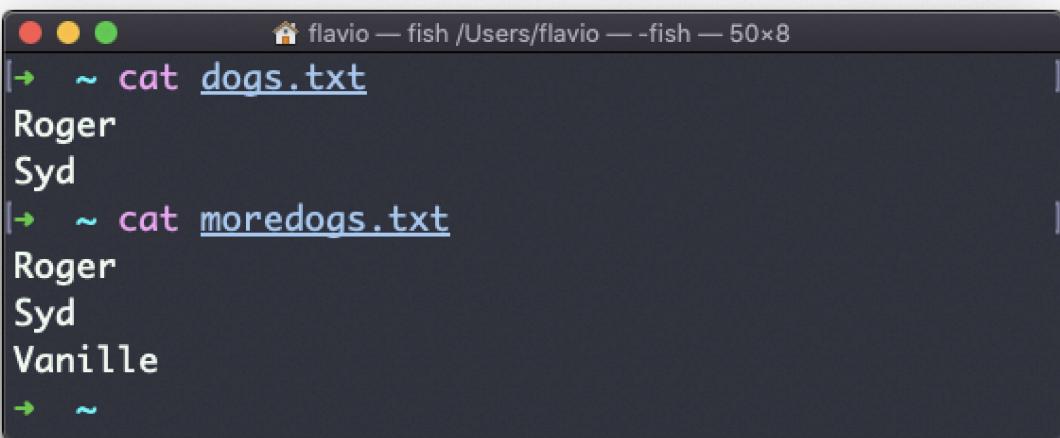
```
flavio — fish /Users/flavio — -fish — 44x8
|~ sort dogs.txt | uniq -c | sort -nr
2 Syd
2 Roger
1 Vanille
1 Tina
1 Luna
1 Ivica
~
```

25. diff

diff é um comando útil. Suponha que você tenha 2 arquivos, que contêm quase as mesmas informações, mas você não consegue encontrar a diferença entre os dois.

diff processará os arquivos e lhe dirá qual é a diferença.

Suponha que você tenha 2 arquivos: e . A diferença é que contém mais um nome de cão:
dogs.txt
moredogs.txt
moredogs.txt



```
flavio — fish /Users/flavio — -fish — 50x8
|~ cat dogs.txt
Roger
Syd
|~ cat moredogs.txt
Roger
Syd
Vanille
~
```

diff dogs.txt moredogs.txt irá dizer-lhe que o segundo arquivo tem mais uma linha, linha 3 com a linha :Vanille



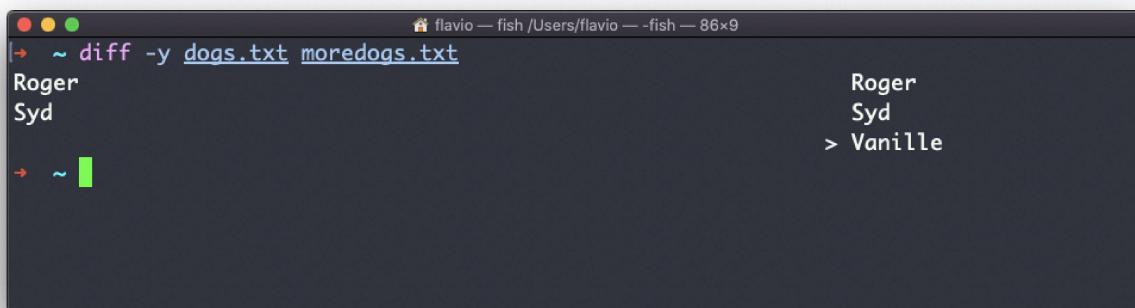
```
flavio — fish /Users/flavio — -fish — 50x5
|~ diff dogs.txt moredogs.txt
2a3
> Vanille
→ ~
```

Se você inverter a ordem dos arquivos, ele informará que o segundo arquivo está faltando na linha 3, cujo conteúdo é:Vanille



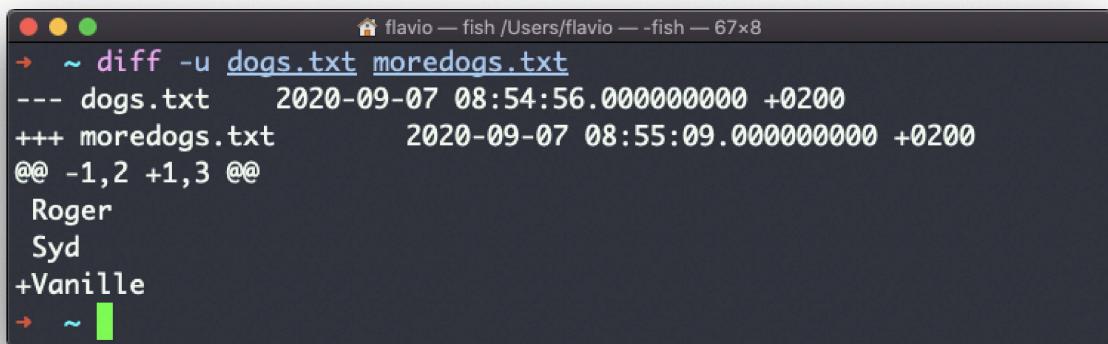
```
flavio — fish /Users/flavio — -fish — 50x5
|~ diff moredogs.txt dogs.txt
3d2
< Vanille
→ ~
```

Usando a opção irá comparar os 2 arquivos linha por linha:-y



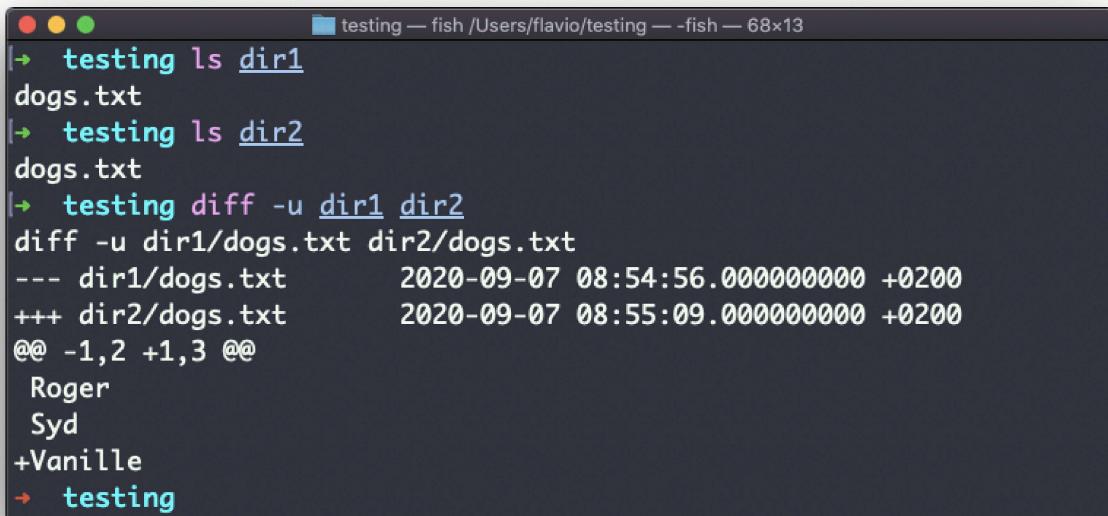
```
flavio — fish /Users/flavio — -fish — 86x9
|~ diff -y dogs.txt moredogs.txt
Roger          Roger
Syd           Syd
              > Vanille
→ ~
```

A opção, no entanto, será mais familiar para você, porque é a mesma usada pelo sistema de controle de versão do Git para exibir diferenças entre as versões:-u



```
flavio — fish /Users/flavio — -fish — 67x8
→ ~ diff -u dogs.txt moredogs.txt
--- dogs.txt      2020-09-07 08:54:56.000000000 +0200
+++ moredogs.txt      2020-09-07 08:55:09.000000000 +0200
@@ -1,2 +1,3 @@
 Roger
 Syd
+Vanille
→ ~
```

A comparação de diretórios funciona da mesma maneira. Você deve usar a opção para comparar recursivamente (indo em subdiretórios):-r



```
testing — fish /Users/flavio/testing — -fish — 68x13
|→ testing ls dir1
dogs.txt
|→ testing ls dir2
dogs.txt
|→ testing diff -u dir1 dir2
diff -u dir1/dogs.txt dir2/dogs.txt
--- dir1/dogs.txt      2020-09-07 08:54:56.000000000 +0200
+++ dir2/dogs.txt      2020-09-07 08:55:09.000000000 +0200
@@ -1,2 +1,3 @@
 Roger
 Syd
+Vanille
→ testing
```

Caso você esteja interessado em quais arquivos diferem, em vez do conteúdo, use as opções e:rq

```
testing — fish /Users/flavio/testing — -fish — 68x5
↳ testing diff -rq dir1 dir2
Files dir1/dogs.txt and dir2/dogs.txt differ
→ testing
```

Há muitas outras opções que você pode explorar na página do manual em execução: `man diff`

```
flavio — man /Users/flavio — less + man diff — 115x55
User Commands
DIFF(1)

NAME
diff — compare files line by line

SYNOPSIS
diff [OPTION]... FILES

DESCRIPTION
Compare files line by line.

-i --ignore-case
    Ignore case differences in file contents.

--ignore-file-name-case
    Ignore case when comparing file names.

--no-ignore-file-name-case
    Consider case when comparing file names.

-E --ignore-tab-expansion
    Ignore changes due to tab expansion.

-b --ignore-space-change
    Ignore changes in the amount of white space.

-w --ignore-all-space
    Ignore all white space.

-B --ignore-blank-lines
    Ignore changes whose lines are all blank.

-I RE --ignore-matching-lines=RE
    Ignore changes whose lines all match RE.

--strip-trailing-cr
    Strip trailing carriage return on input.

-a --text
    Treat all files as text.

-c -C NUM --context[=NUM]
    Output NUM (default 3) lines of copied context.

-u -U NUM --unified[=NUM]
    Output NUM (default 3) lines of unified context.

--label LABEL
    Use LABEL instead of file name.

-p --show-c-function
    Show which C function each change is in.

-F RE --show-function-line=RE
```

26. echo

O comando faz um trabalho simples: imprime na saída o argumento passado para ela.echo

Este exemplo:

```
echo "hello"
```

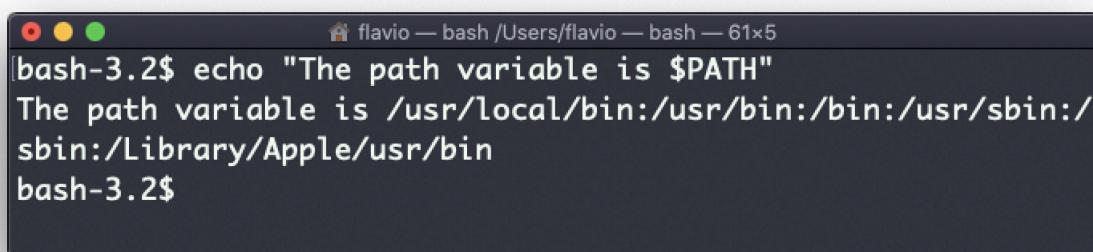
imprimirá no terminal.hello

Podemos anexar a saída a um arquivo:

```
echo "hello" >> output.txt
```

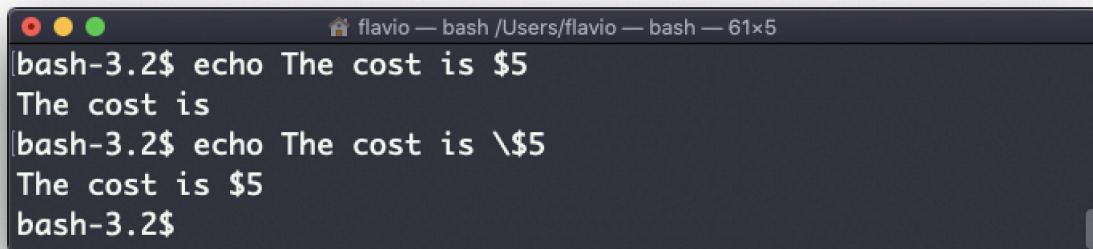
Podemos interpolar variáveis de ambiente:

```
echo "The path variable is $PATH"
```



A screenshot of a macOS terminal window titled "flavio — bash /Users/flavio — bash — 61x5". The window shows the command "bash-3.2\$ echo "The path variable is \$PATH"" being run, followed by the output "The path variable is /usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin". The window has the standard OS X title bar with red, yellow, and green buttons.

Cuidado que os caracteres especiais precisam ser escapados com uma barra invertida. por exemplo:\\$



```
flavio — bash /Users/flavio — bash — 61x5
|bash-3.2$ echo The cost is $5
|The cost is
|bash-3.2$ echo The cost is \$5
|The cost is $5
|bash-3.2$
```

Este é apenas o começo. Podemos fazer algumas coisas boas quando se trata de interagir com os recursos do shell.

Podemos ecoar os arquivos na pasta atual:

```
echo *
```

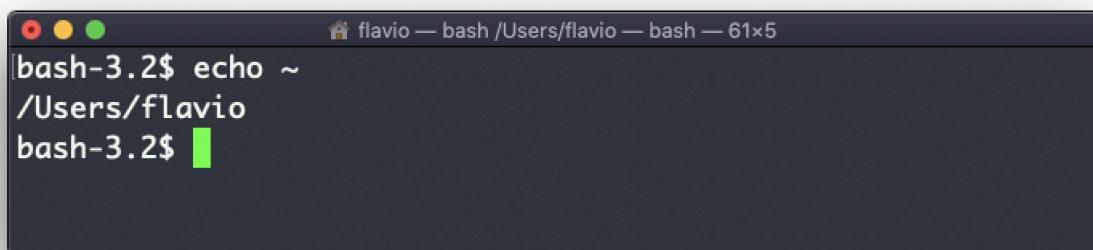
Podemos ecoar os arquivos na pasta atual que começam com a letra: o

```
echo o*
```

Qualquer comando e recurso válido do Bash (ou qualquer shell que você esteja usando) pode ser usado aqui.

Você pode imprimir o caminho da pasta base:

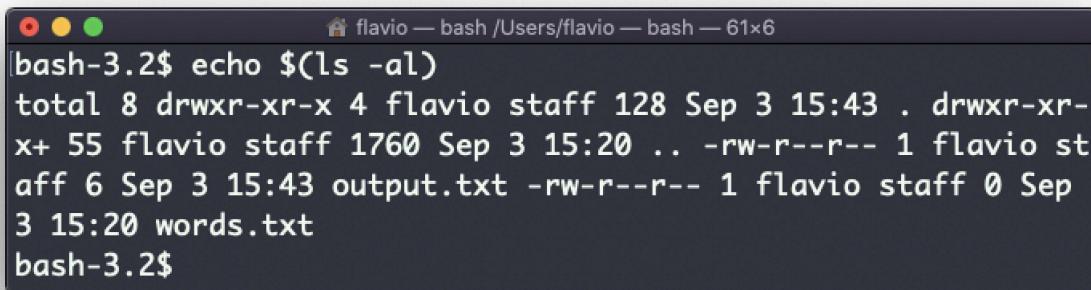
```
echo ~
```



```
flavio — bash /Users/flavio — bash — 61x5
|bash-3.2$ echo ~
|/Users/flavio
|bash-3.2$
```

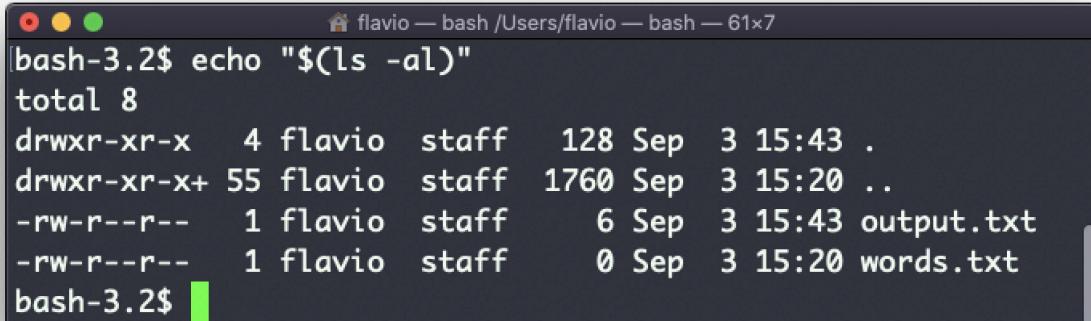
Você também pode executar comandos e imprimir o resultado na saída padrão (ou no arquivo, como viu):

```
echo $(ls -al)
```



```
flavio — bash /Users/flavio — bash — 61x6
bash-3.2$ echo $(ls -al)
total 8 drwxr-xr-x 4 flavio staff 128 Sep 3 15:43 .
drwxr-xr-x+ 55 flavio staff 1760 Sep 3 15:20 ..
-rw-r--r-- 1 flavio staff 6 Sep 3 15:43 output.txt
-rw-r--r-- 1 flavio staff 0 Sep 3 15:20 words.txt
bash-3.2$
```

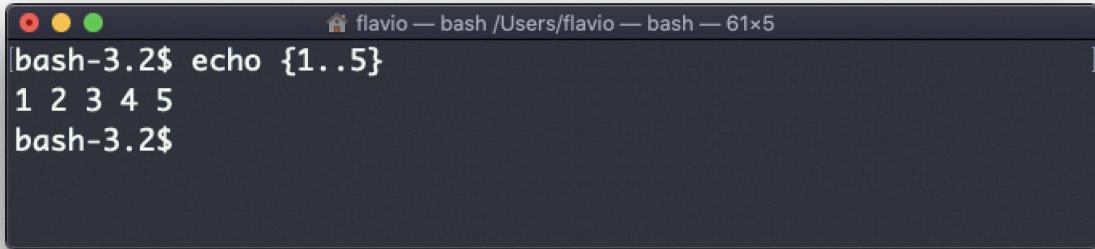
Observe que o espaço em branco não é preservado por padrão. Você precisa envolver o comando entre aspas duplas para fazer isso:



```
flavio — bash /Users/flavio — bash — 61x7
bash-3.2$ echo "$(ls -al)"
total 8
drwxr-xr-x 4 flavio staff 128 Sep 3 15:43 .
drwxr-xr-x+ 55 flavio staff 1760 Sep 3 15:20 ..
-rw-r--r-- 1 flavio staff 6 Sep 3 15:43 output.txt
-rw-r--r-- 1 flavio staff 0 Sep 3 15:20 words.txt
bash-3.2$
```

Você pode gerar uma lista de cadeias de caracteres, por exemplo, intervalos:

```
echo {1..5}
```



```
fladio — bash /Users/fladio — bash — 61x5
bash-3.2$ echo {1..5}
1 2 3 4 5
bash-3.2$
```

27. chown

Cada arquivo/diretório em um sistema operacional como Linux ou macOS (e todos os sistemas UNIX em geral) tem um **proprietário**.

O proprietário de um arquivo pode fazer tudo com ele. Ele pode decidir o destino desse arquivo.

O proprietário (e o usuário) também pode alterar o proprietário para outro usuário, usando o comando:rootchown

```
chown <owner> <file>
```

Assim:

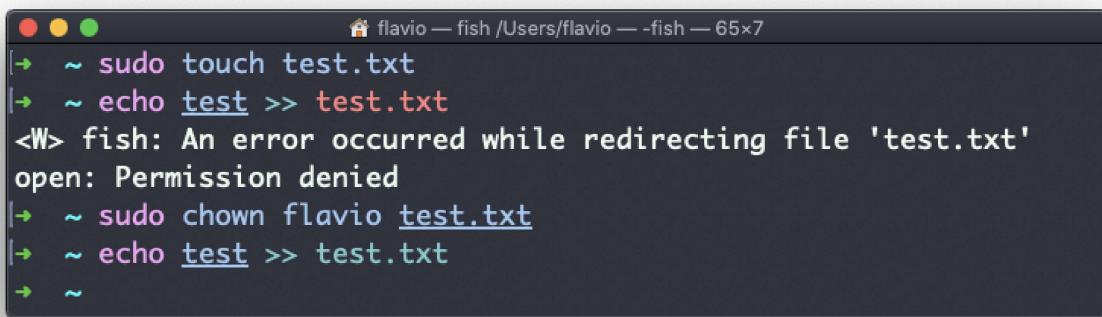
```
chown fladio test.txt
```

Por exemplo, se você tiver um arquivo de propriedade do , não poderá gravar nele como outro usuário:root



```
flavo — fish /Users/flavio — -fish — 65x7
|~ sudo touch test.txt
|~ echo test >> test.txt
<W> fish: An error occurred while redirecting file 'test.txt'
open: Permission denied
|~ ~
```

Você pode usar para transferir a propriedade para você:`chown`



```
flavo — fish /Users/flavio — -fish — 65x7
|~ sudo touch test.txt
|~ echo test >> test.txt
<W> fish: An error occurred while redirecting file 'test.txt'
open: Permission denied
|~ sudo chown flavio test.txt
|~ echo test >> test.txt
|~ ~
```

É bastante comum ter a necessidade de alterar a propriedade de um diretório e, recursivamente, todos os arquivos contidos, além de todos os subdiretórios e os arquivos contidos neles também.

Você pode fazer isso usando o sinalizador:`-R`

```
chown -R <owner> <file>
```

Arquivos/diretórios não têm apenas um proprietário, eles também têm um **grupo**. Através deste comando você pode alterar isso simultaneamente enquanto você muda o proprietário:

```
chown <owner>:<group> <file>
```

Exemplo:

```
chown flavio:users test.txt
```

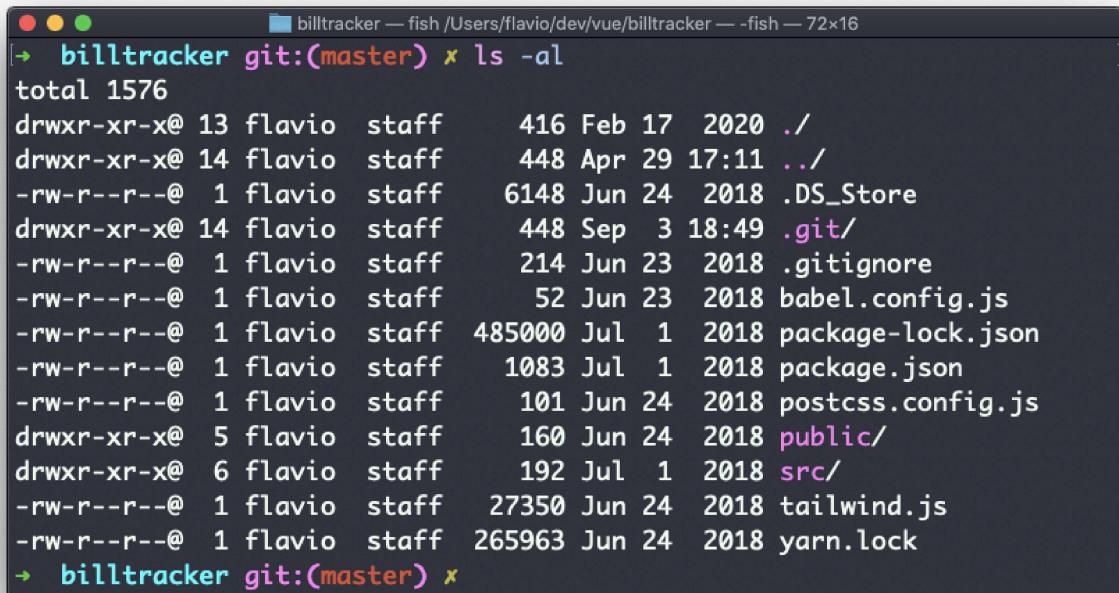
Você também pode simplesmente alterar o grupo de um arquivo usando o comando:chgrp

```
chgrp <group> <filename>
```

28. chmod

Cada arquivo nos sistemas operacionais Linux / macOS (e sistemas UNIX em geral) tem 3 permissões: Ler, gravar, executar.

Vá para uma pasta e execute o comando.ls -al



```
billtracker — fish /Users/flavio/dev/vue/billtracker — -fish — 72x16
[→ billtracker git:(master) ✘ ls -al
total 1576
drwxr-xr-x@ 13 flavio  staff   416 Feb 17  2020 .
drwxr-xr-x@ 14 flavio  staff   448 Apr 29 17:11 ..
-rw-r--r--@  1 flavio  staff  6148 Jun 24  2018 .DS_Store
drwxr-xr-x@ 14 flavio  staff   448 Sep  3 18:49 .git/
-rw-r--r--@  1 flavio  staff   214 Jun 23  2018 .gitignore
-rw-r--r--@  1 flavio  staff    52 Jun 23  2018 babel.config.js
-rw-r--r--@  1 flavio  staff  485000 Jul  1  2018 package-lock.json
-rw-r--r--@  1 flavio  staff   1083 Jul  1  2018 package.json
-rw-r--r--@  1 flavio  staff    101 Jun 24  2018 postcss.config.js
drwxr-xr-x@  5 flavio  staff   160 Jun 24  2018 public/
drwxr-xr-x@  6 flavio  staff   192 Jul  1  2018 src/
-rw-r--r--@  1 flavio  staff  27350 Jun 24  2018 tailwind.js
-rw-r--r--@  1 flavio  staff  265963 Jun 24  2018 yarn.lock
→ billtracker git:(master) ✘
```

As cadeias de caracteres estranhas que você vê em cada linha de arquivo, como , definem as permissões do arquivo ou pasta.drwxr-xr-x

Vamos dissecá-lo.

A primeira letra indica o tipo de arquivo:

- - significa que é um arquivo normal
- d significa que é um diretório
- l significa que é um link

Então você tem 3 conjuntos de valores:

- O primeiro conjunto representa as permissões do **proprietário** do arquivo
- O segundo conjunto representa as permissões dos membros do grupo ao **qual** o arquivo está associado
- O terceiro conjunto representa as permissões de **todos os outros**

Esses conjuntos são compostos por 3 valores. significa que *uma persona* específica tem acesso de leitura, gravação e execução. Qualquer coisa que é removida é trocada por um , que permite formar várias combinações de valores e permissões relativas: , , , e assim por diante.rwx-rw-r--r-x

Você pode alterar as permissões dadas a um arquivo usando o comando chmod

chmod pode ser usado de 2 maneiras. O primeiro é usar argumentos simbólicos, o segundo é usar argumentos numéricos. Vamos começar com os símbolos primeiro, o que é mais intuitivo.

Digite seguido por um espaço e uma letra:chmod

- a representa *todos*
- u significa *usuário*
- g significa *grupo*
- o significa *outros*

Em seguida, digite ou para adicionar uma permissão, ou para removê-la. Em seguida, insira um ou mais símbolos de permissões (,).+-rwx

Tudo seguido pelo nome do arquivo ou pasta.

Aqui estão alguns exemplos:

```
chmod a+r filename #everyone can now read  
chmod a+rw filename #everyone can now read and write  
chmod o-rwx filename #others (not the owner, not in the same group of  
the file) cannot read, write or execute the file
```

Você pode aplicar as mesmas permissões a várias personas adicionando várias letras antes do /:+-

```
chmod og-r filename #other and group can't read any more
```

Caso esteja editando uma pasta, você pode aplicar as permissões a todos os arquivos contidos nessa pasta usando o sinalizador (recursivo).-r

Os argumentos numéricos são mais rápidos, mas acho difícil lembrá-los quando você não os está usando no dia a dia. Você usa um dígito que representa as permissões da persona. Esse valor numérico pode ser no máximo 7 e é calculado da seguinte maneira:

- 1 se tiver permissão de execução
- 2 se tiver permissão de gravação
- 4 se tiver permissão de leitura

Isso nos dá 4 combinações:

- 0 sem permissões
- 1 pode executar
- 2 pode escrever
- 3 pode escrever, executar
- 4 pode ler
- 5 pode ler, executar
- 6 pode ler, escrever
- 7 pode ler, escrever e executar

Nós os usamos em pares de 3, para definir as permissões de todos os 3 grupos ao todo:

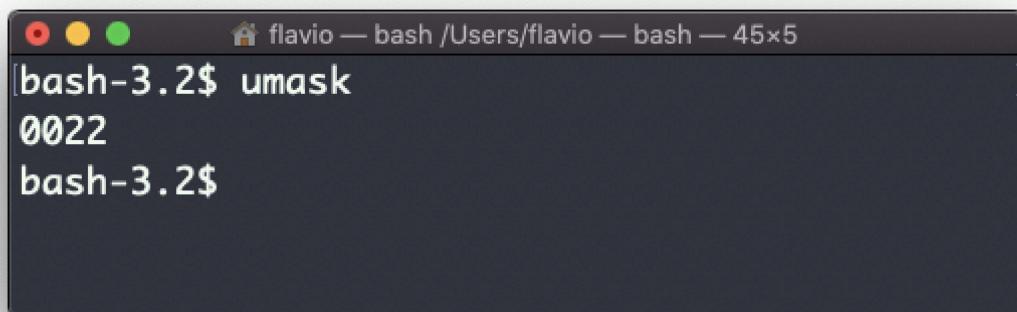
```
chmod 777 filename  
chmod 755 filename  
chmod 644 filename
```

29. umask

Ao criar um arquivo, você não precisa decidir as permissões antecipadamente. As permissões têm padrões.

Esses padrões podem ser controlados e modificados usando o comando `umask`

Digitar sem argumentos mostrará o `umask` atual, neste caso: `umask 0022`

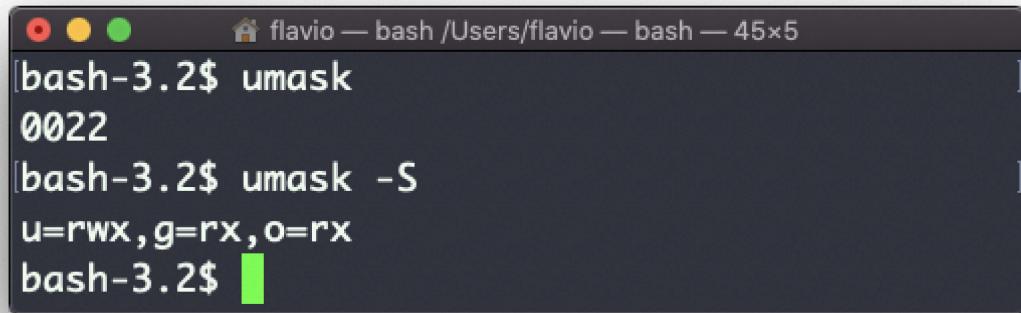


The screenshot shows a dark-themed macOS terminal window. At the top, it displays the user 'flavio' at the path '/Users/flavio' in a bash shell, with a window size of 45x5. The main text area contains the command 'bash-3.2\$ umask' followed by the output '0022'. Below this, another prompt 'bash-3.2\$' is visible. The window has standard OS X window controls (red, yellow, green buttons) in the top-left corner.

O que significa? Esse é um valor octal que representa as permissões. `0022`

Outro valor comum é `.0002`

Use para ver uma notação legível por humanos: `umask -S`



```
flavio — bash /Users/flavio — bash — 45x5
[bash-3.2$ umask
0022
[bash-3.2$ umask -S
u=rwx,g=rx,o=rx
bash-3.2$ ]]
```

Nesse caso, o usuário (), proprietário do arquivo, tem permissões de leitura, gravação e execução em arquivos.u

Outros usuários pertencentes ao mesmo grupo () têm permissão de leitura e execução, igual a todos os outros usuários ().go

Na notação numérica, normalmente alteramos os últimos 3 dígitos.

Aqui está uma lista que dá um significado ao número:

- 0 ler, gravar, executar
- 1 ler e escrever
- 2 ler e executar
- 3 somente leitura
- 4 escrever e executar
- 5 somente gravação
- 6 executar somente
- 7 sem permissões

Observe que essa notação numérica difere da que usamos em .chmod

Podemos definir um novo valor para a máscara definindo o valor em formato numérico:

```
umask 002
```

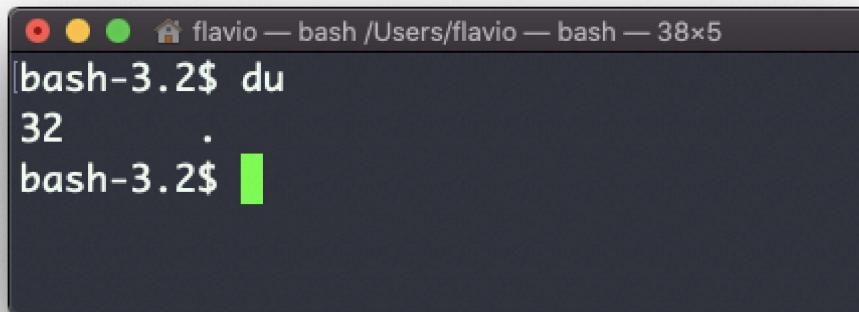
ou você pode alterar a permissão de uma função específica:

```
umask g+r
```

30. du

O comando calculará o tamanho de um diretório como um todo:du

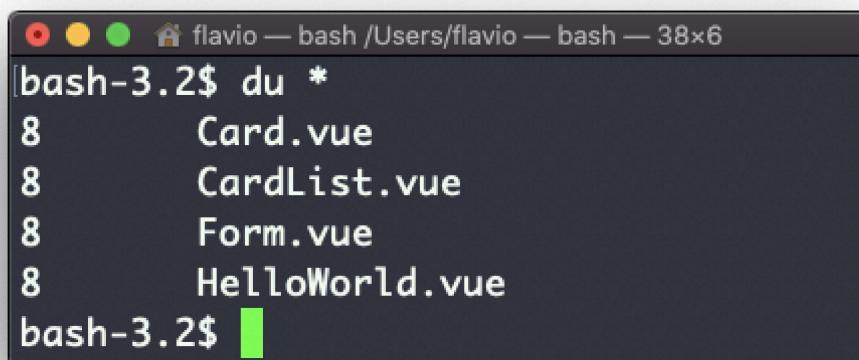
```
du
```



```
[bash-3.2$ du
32 .
bash-3.2$ ]
```

O número aqui é um valor expresso em bytes.32

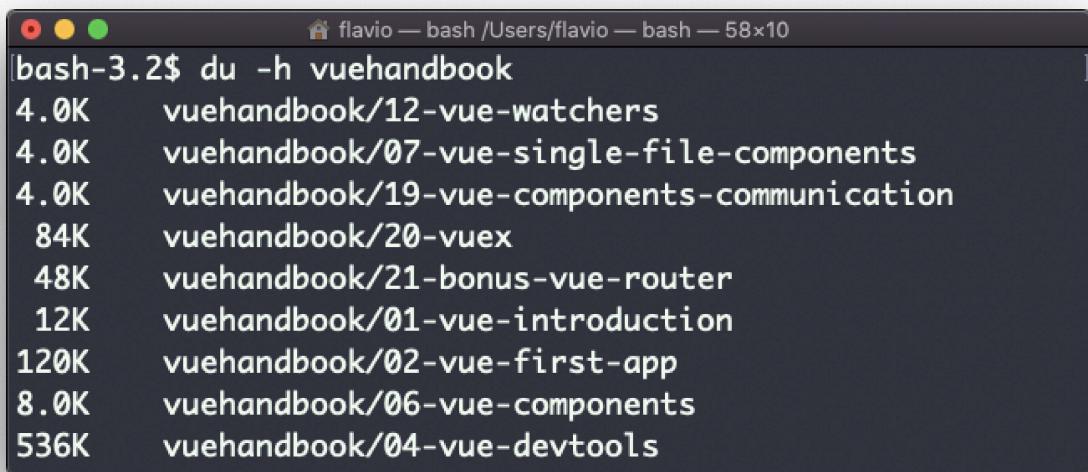
A execução calculará o tamanho de cada arquivo individualmente:du *



```
[bash-3.2$ du *
8 Card.vue
8 CardList.vue
8 Form.vue
8 HelloWorld.vue
bash-3.2$ ]
```

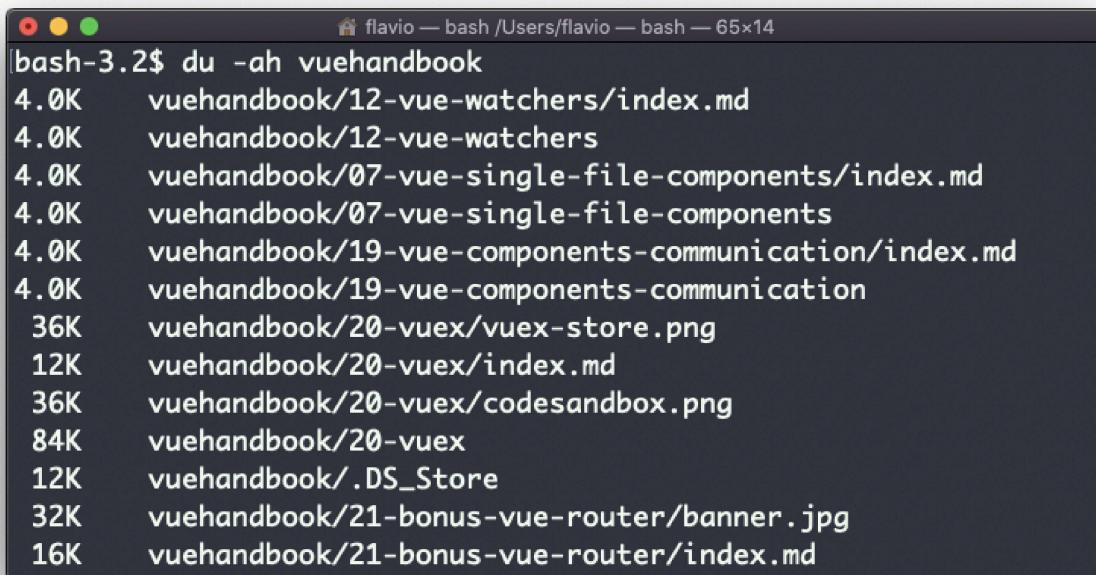
Você pode definir para exibir valores em MegaBytes usando o `m`, e GigaBytes usando o `.dudu -mdu -g`

A opção mostrará uma notação legível por humanos para tamanhos, adaptando-se ao tamanho:`-h`



```
flavio — bash /Users/flavio — bash — 58x10
[bash-3.2$ du -h vuehandbook
4.0K    vuehandbook/12-vue-watchers
4.0K    vuehandbook/07-vue-single-file-components
4.0K    vuehandbook/19-vue-components-communication
84K     vuehandbook/20-vuex
48K     vuehandbook/21-bonus-vue-router
12K     vuehandbook/01-vue-introduction
120K    vuehandbook/02-vue-first-app
8.0K    vuehandbook/06-vue-components
536K    vuehandbook/04-vue-devtools
```

Adicionar a opção também imprimirá o tamanho de cada arquivo nos diretórios:`-a`

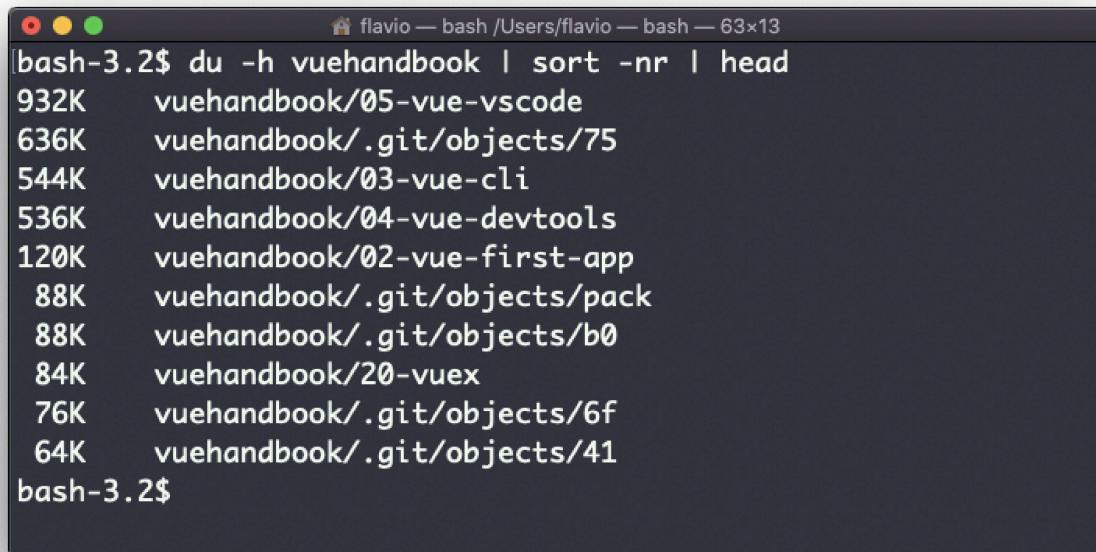


```
flavio — bash /Users/flavio — bash — 65x14
[bash-3.2$ du -ah vuehandbook
4.0K    vuehandbook/12-vue-watchers/index.md
4.0K    vuehandbook/12-vue-watchers
4.0K    vuehandbook/07-vue-single-file-components/index.md
4.0K    vuehandbook/07-vue-single-file-components
4.0K    vuehandbook/19-vue-components-communication/index.md
4.0K    vuehandbook/19-vue-components-communication
36K     vuehandbook/20-vuex/vuex-store.png
12K     vuehandbook/20-vuex/index.md
36K     vuehandbook/20-vuex/codesandbox.png
84K     vuehandbook/20-vuex
12K     vuehandbook/.DS_Store
32K     vuehandbook/21-bonus-vue-router/banner.jpg
16K     vuehandbook/21-bonus-vue-router/index.md
```

Uma coisa útil é classificar os diretórios por tamanho:

```
du -h <directory> | sort -nr
```

e, em seguida, canalização para obter apenas os primeiros 10 resultados:head

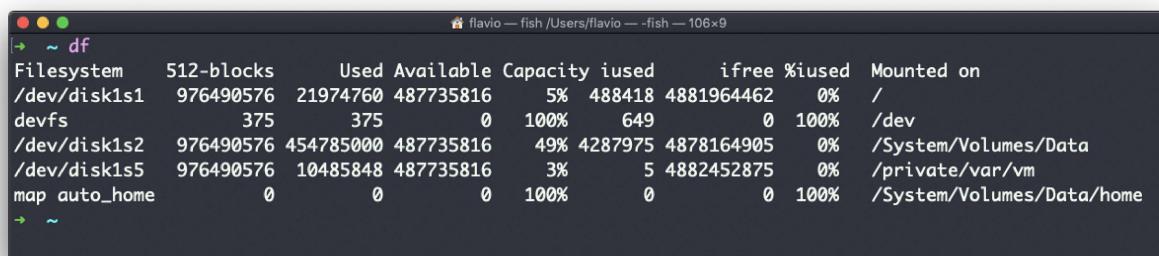


```
flavio — bash /Users/flavio — bash — 63x13
[bash-3.2$ du -h vuehandbook | sort -nr | head
932K    vuehandbook/05-vue-vscode
636K    vuehandbook/.git/objects/75
544K    vuehandbook/03-vue-cli
536K    vuehandbook/04-vue-devtools
120K    vuehandbook/02-vue-first-app
88K     vuehandbook/.git/objects/pack
88K     vuehandbook/.git/objects/b0
84K     vuehandbook/20-vuex
76K     vuehandbook/.git/objects/6f
64K     vuehandbook/.git/objects/41
bash-3.2$
```

31. df

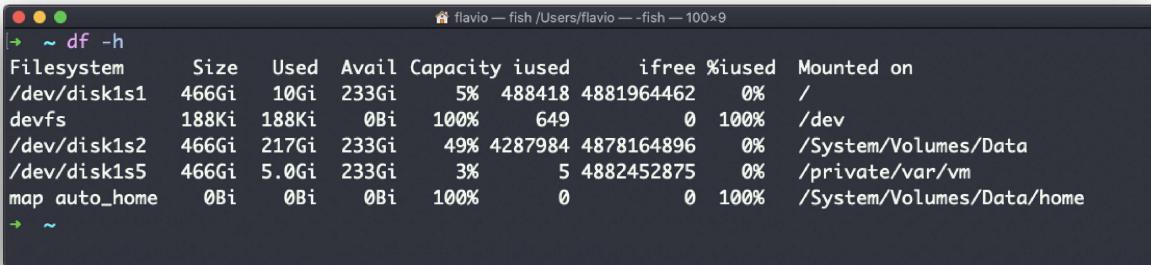
O comando é usado para obter informações de uso do disco.df

Seu formulário básico imprimirá informações sobre os volumes montados:



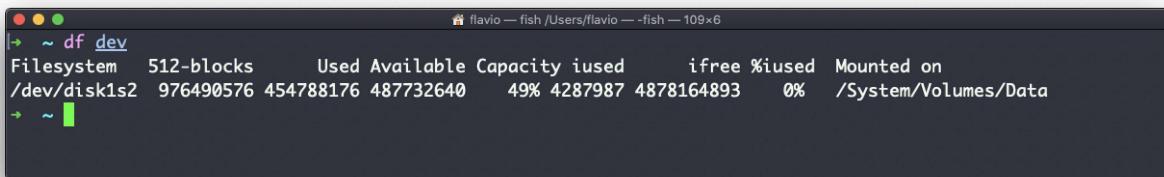
```
flavio — fish /Users/flavio — fish — 106x9
[~] ~ df
Filesystem 512-blocks Used Available Capacity iused ifree %iused Mounted on
/dev/disk1s1 976490576 21974760 487735816 5% 488418 4881964462 0% /
devfs      375    375     0 100%   649      0 100% /dev
/dev/disk1s2 976490576 454785000 487735816 49% 4287975 4878164905 0% /System/Volumes/Data
/dev/disk1s5 976490576 10485848 487735816 3%   5 4882452875 0% /private/var/vm
map auto_home      0      0     0 100%     0      0 100% /System/Volumes/Data/home
[~]
```

Using the option (-h) will show those values in a human-readable format:-hdf -h



```
flavio — fish /Users/flavio — -fish — 100x9
[~] ~ df -h
Filesystem      Size   Used  Avail Capacity iused ifree %iused Mounted on
/dev/disk1s1    466Gi  10Gi  233Gi    5%  488418 4881964462  0%   /
devfs          188Ki  188Ki  0Bi   100%   649     0  100%   /dev
/dev/disk1s2    466Gi  217Gi  233Gi   49%  4287984 4878164896  0%   /System/Volumes/Data
/dev/disk1s5    466Gi  5.0Gi  233Gi   3%   5 4882452875  0%   /private/var/vm
map auto_home   0Bi   0Bi   0Bi   100%   0       0  100%   /System/Volumes/Data/home
[~] ~
```

Você também pode especificar um nome de arquivo ou diretório para obter informações sobre o volume específico em que ele vive:



```
flavio — fish /Users/flavio — -fish — 109x6
[~] ~ df dev
Filesystem 512-blocks   Used Available Capacity iused ifree %iused Mounted on
/dev/disk1s2 976490576 454788176 487732640  49% 4287987 4878164893  0%   /System/Volumes/Data
[~] ~
```

32. nome base

Suponha que você tenha um caminho para um arquivo, por exemplo./Users/flavio/test.txt

Executando

```
basename /Users/flavio/test.txt
```

retornará a cadeia de caracteres: test.txt



```
fladio — fish /Users/flavio — -fish — 42x5
→ ~ basename /Users/flavio/test.txt
test.txt
→ ~ |
```

Se você executar em uma cadeia de caracteres de caminho que aponta para um diretório, você obterá o último segmento do caminho. Neste exemplo, há um diretório:basename/Users/flavio



```
fladio — fish /Users/flavio — -fish — 42x5
|→ ~ basename /Users/flavio/
flavio
|→ ~ basename /Users/flavio
flavio
→ ~ |
```

33. dirname

Suponha que você tenha um caminho para um arquivo, por exemplo./Users/flavio/test.txt

Executando

```
dirname /Users/flavio/test.txt
```

retornará a cadeia de caracteres:/Users/flavio

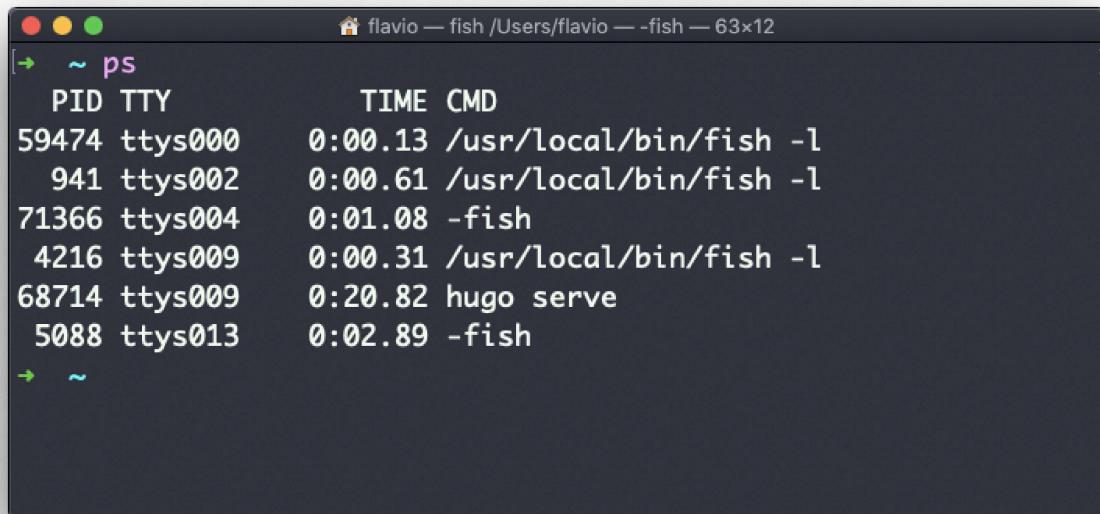


```
flavio — fish /Users/flavio — -fish — 44x5
[→ ~ dirname /Users/flavio/test.txt
/Users/flavio
→ ~
```

34. ps

Seu computador está executando, em todos os momentos, toneladas de processos diferentes.

Você pode inspecioná-los todos usando o comando:ps



```
flavio — fish /Users/flavio — -fish — 63x12
[→ ~ ps
 PID TTY          TIME CMD
 59474 ttys000    0:00.13 /usr/local/bin/fish -l
   941 ttys002    0:00.61 /usr/local/bin/fish -l
 71366 ttys004    0:01.08 -fish
 4216 ttys009    0:00.31 /usr/local/bin/fish -l
 68714 ttys009    0:20.82 hugo serve
  5088 ttys013    0:02.89 -fish
→ ~
```

Esta é a lista de processos iniciados pelo usuário atualmente em execução na sessão atual.

Aqui eu tenho algumas instâncias de shell, a maioria aberta pelo VS Code dentro do editor, e uma instância do Hugo executando a visualização de desenvolvimento de um site.fish