Comandos úteis no Terminal Linux (Parte 1)

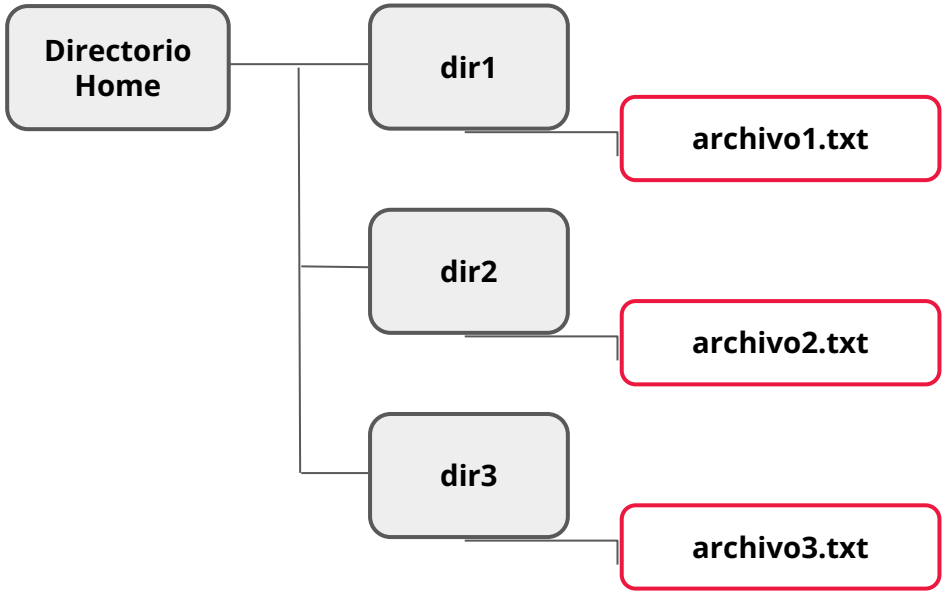
DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 Preparação do ambiente

Consolidando nosso ambiente



Para seguir corretamente os exemplos subsequentes, é desejável que seus ambientes (máquina virtual ou WSL) tenham a seguinte estrutura de pastas e arquivos replicada.

Consolidando nosso ambiente

Para isso, devemos executar os seguintes comandos, em uma determinada ordem (algum ou texto que seja após ou prompt, ou seja, após ou "$", em branco):

```
edorio@DESKTOP-W10:~$ mkdir dir1 dir2 dir3
edorio@DESKTOP-W10:~$ touch dir1/archivo1.txt
dir2/archivo2.txt dir3/archivo3.txt
```

Verificando nosso ambiente

Listamos os diretórios com a instrução "ls -R", obtendo o seguinte:

```
edorio@DESKTOP-W10:~$ ls -R
.:
dir1  dir2  dir3
./dir1:
archivo1.txt
./dir2:
archivo2.txt
./dir3:
archivo3.txt
```

2 Comandos para manipulação de arquivos

ls (lista de diretórios e arquivos)

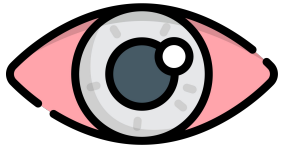
Com o comando **ls** você poderá listar os diferentes arquivos e diretórios da pasta de trabalho em que se encontra. O comando aceita uma infinidade de opções, algumas das quais veremos a seguir.

Na imagem abaixo podemos ver o uso mais simples do comando ls. Se não dermos nenhuma opção, ele listará todos os arquivos e diretórios encontrados na pasta de trabalho atual, independentemente dos arquivos ocultos.

```
edorio@DESKTOP-W10:~$ ls
dir1  dir2  dir3
```

ls (lista de diretórios e arquivos)

Opção -a: Com esta opção, o comando mostrará a você, na forma de uma lista, todo o conteúdo do diretório de trabalho, incluindo arquivos e pastas ocultos. Dependendo do shell, alguns tipos de arquivo serão exibidos em cores diferentes.



```
edorio@DESKTOP-W10:~$ ls -a
.          .aws            .bash_logout   .config     .landscape
.profile                    .vagrant.d  dir3 ..          .azure
.bashrc         .docker     .local         .ssh     dir1 .ansible
.bash_history   .cache          .fastlane   .motd_shown
.sudo_as_admin_successful   dir2
```

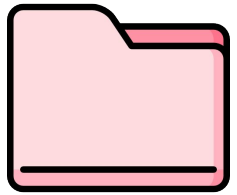
ls (lista de diretórios e arquivos)

Opção -l: Esta opção é semelhante a “-a”, mas mostra o conteúdo como uma lista e inclui informações sobre cada elemento. É um dos mais utilizados, sendo especialmente útil quando se trata de conhecer o proprietário e as permissões de cada arquivo.

```
edorio@DESKTOP-W10:~$ ls -l
total 12
drwxr-xr-x 2 edorio edorio 4096 May 21 01:59 dir1
drwxr-xr-x 2 edorio edorio 4096 May 21 01:59 dir2
drwxr-xr-x 2 edorio edorio 4096 May 21 01:59 dir3
```

mkdir (criar diretórios)

O comando mkdir permitirá que você crie um diretório com o nome e o caminho que você especificar. Se você não indicar nenhum caminho, por padrão, ele criará a pasta dentro do diretório de trabalho em que você se encontra.



```
{}  edorio@DESKTOP-W10:~$ mkdir dir4
```

Caso contrário, podemos dizer a ele para criar um diretório para nós com um caminho definido em dir1.

```
{}  edorio@DESKTOP-W10:~$ mkdir dir1/subdir1
```

rmdir (excluir diretórios)

O comando rmdir permite excluir o diretório especificado. Para usar este comando, o diretório a ser excluído deve estar vazio.

```
{}  edorio@DESKTOP-W10:~$ rmdir dir4
```

O acima é o uso mais simples do comando, sem indicar o caminho.

Também podemos excluir um diretório com um caminho definido.

```
{}  edorio@DESKTOP-W10:~$ rmdir dir1/subdir1
```

rm (Excluir diretórios e arquivos não vazios)

O comando rm permite remover arquivos e diretórios individuais que não estão vazios.

```
{}  edorio@DESKTOP-W10:~$ rm dir1/archivo1.txt
```

O acima é o uso mais simples do comando, sem indicar o caminho.

Excluímos 1 arquivo específico dentro de dir1.

```
{}  edorio@DESKTOP-W10:~$ rm -r dir2
```

Com o modificador -r excluímos o diretório dir2 e recursivamente, todo o seu conteúdo. É um comando a ser usado com muita cautela.

cp (Copiar arquivos e diretórios)

Usando o **comando cp**, você poderá copiar arquivos e diretórios, bem como colocá-los em outros caminhos, definindo primeiro a origem e depois o destino.

```
{}  edorio@DESKTOP-W10:~$ cp dir3/archivo3.txt dir1/archivo1.txt
```

O acima é o uso mais simples do comando, sem indicar o caminho.

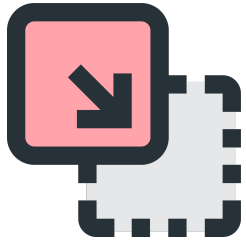
Neste caso, copiamos o arquivo3.txt para o dir1 e o nomeamos como arquivo1.txt

```
{}  edorio@DESKTOP-W10:~$ cp -r dir3 dir2
```

Com o modificador -r neste caso copiamos o diretório dir3 em um chamado dir2, o mesmo comando o cria.

mv (Mover arquivos e diretórios)

O comando mv o ajudará a mover arquivos do console. A sintaxe é muito simples, você só precisa especificar o local inicial, incluindo o nome do arquivo e o local de destino.



```
{}  edorio@DESKTOP-W10:~$ mv dir1/archivo1.txt dir3/archivo1.txt
```

Movemos um arquivo de dir1 para dir3, mantendo seu nome original.

```
{}  edorio@DESKTOP-W10:~$ mv dir3/archivo1.txt dir3/archivo3bis.txt
```

Neste caso usamos o comando mv para renomear um arquivo, pois os caminhos definidos são os mesmos.

3 Comandos para ler arquivos de texto

cat (Ler e modificar arquivos)

O comando cat é um dos comandos mais usados quando

tenta manipular arquivos de texto (no formato .txt) do terminal.

Entre suas muitas opções, está a possibilidade de criar um arquivo, imprimindo seu conteúdo na tela.

```
{}  edorio@DESKTOP-W10:~$ cat >dir1/archivo1.txt
```

Isso abrirá o arquivo1.txt, permitindo-nos editá-lo. Com a combinação CTRL + D finalizaremos a edição e o conteúdo será salvo.

```
{}  edorio@DESKTOP-W10:~$ cat dir1/archivo1.txt
    Hola Digital House, esto es CAT
```

Invocando o comando sem ">", ele nos mostrará seu conteúdo na tela. Pode ser usado com o modificador -n, para numerar as linhas e com o -b, para não mostrar linhas em branco.

mais (ler arquivos)

O comando more é outro comando útil para imprimir em tela o conteúdo de um arquivo de texto. É essencialmente o mesmo que o comando cat, com a diferença de que este comando página o conteúdo e é mais adequado para ler arquivos longos.

```
{}  edorio@DESKTOP-W10:~$ more /var/log/dpkg.log
```

Ele irá paginar o arquivo em questão, de tal forma que em suas últimas linhas o veremos assim:

```
2021-02-19 23:56:28 remove linux-headers-5.4.0-65:all 5.4.0-65.73 <none>
2021-02-19 23:56:28 status half-configured linux-headers-5.4.0-65:all 5.4.0-65.73
2021-02-19 23:56:28 status half-installed linux-headers-5.4.0-65:all 5.4.0-65.73
--More--(5%)
```

nano (editor de texto no terminal)

Nano é um editor de texto para o terminal, que mais do que ler arquivos serve para modificá-los e editá-los, embora para este guia também seja perfeito para abrirmos o arquivo e visualizar seu conteúdo a partir da linha de comando:

```
{}  edorio@DESKTOP-W10:~$ nano dir1/archivo1.txt
```

Vai nos mostrar:

```
  GNU nano 4.8                                    dir1/archivo1.txt
Hola Digital House, esto es CAT
```

Uma vez aberta, a parte inferior exibirá as diferentes combinações de teclas que você precisará ao trabalhar com arquivos.

nano (editor de texto no terminal)

A parte inferior mostra as diferentes combinações de teclas que serão necessárias ao trabalhar com arquivos.

CTRL+R: Combinação para indicar um arquivo de texto ao Nano para abrir e exibir seu conteúdo no console.

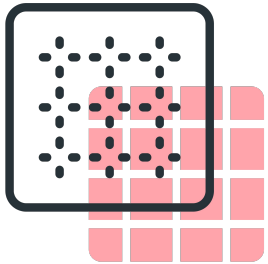
CTRL+V: Estando dentro do Nano e com o arquivo aberto no console, essa combinação serve para avançar para a próxima página.

CTRL+Y: Usado para voltar à página anterior.

CTRL+W: Usado para inserir um caractere ou grupo de caracteres e para pesquisar no texto qualquer letra ou palavra que corresponda ao parâmetro de pesquisa.

CTRL+X: Para fechar o arquivo após terminar de visualizá-lo no console. Isso fechará o editor de texto Nano e o prompt Bash do console reaparecerá.

grep (procure um padrão nos arquivos)



O comando grep, pertencente à família Unix, é uma das ferramentas mais versáteis e úteis disponíveis. Isso procura um padrão que definimos em um arquivo de texto. Seu primeiro parâmetro é a string de texto a ser pesquisada, então no(s) arquivo(s) (aceita curingas como *, podendo ir recursivamente com o modificador -r) que vamos pesquisar.

```
{}  edorio@DESKTOP-W10:~$ grep "Digital House" * -r
```

Neste caso procuramos a string "Digital House" em todos os arquivos, recursivamente, a execução retornou o seguinte:

```
edorio@AR-CARRERA-09:~$ grep "Digital House" * -r
dir1/archivo1.txt:Hola Digital House, esto es CAT
edorio@AR-CARRERA-09:~$
```

tee (Leia uma entrada e escreva-a)

O comando tee do Linux lê a entrada padrão e a grava na saída padrão e em um ou mais arquivos. Normalmente, no redirecionamento de saída, as linhas de comando são escritas em um arquivo, mas se quisermos ver essa saída ao mesmo tempo, não podemos. Usando o comando tee, é possível alcançá-lo!

```
{}  edorio@DESKTOP-W10:~$ ls -l | tee listado.txt
```

Nesse caso, além de nos mostrar o diretório, ele será salvo em um arquivo.

```
{}  edorio@DESKTOP-W10:~$ ls -l | tee -a listado.txt
```

Usando a opção -a, o conteúdo é adicionado ao arquivo, sem pisar no acima.


DigitalHouse>
Coding School