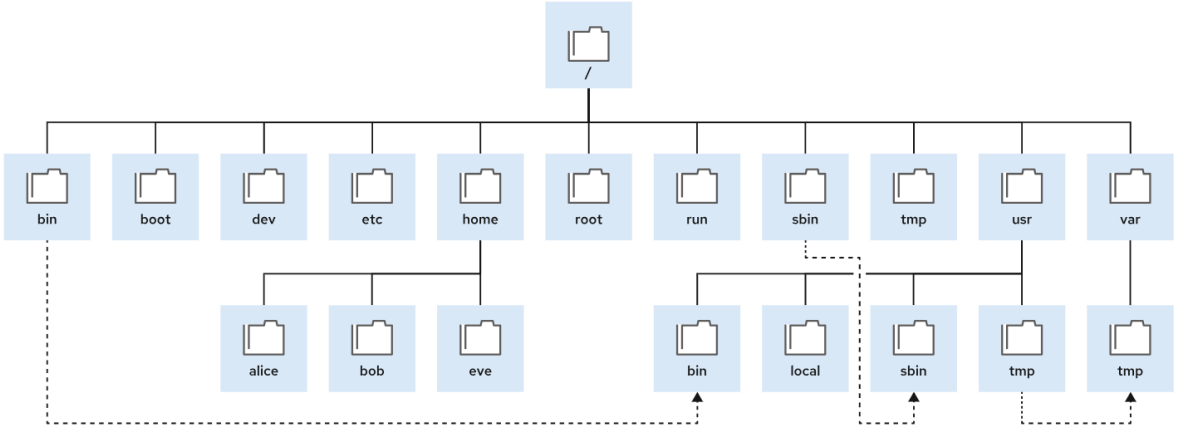
Capitulo 3. Gerenciar arquivos a partir da linha de comando

A hierarquia o sistema de arquivos

O sistema Linux armazena todos os arquivos em sistemas de arquivos, que são organizados em uma única árvore de diretório invertida, conhecida como hierarquia do sistema de arquivos.

Essa hierarquia é uma árvore invertida porque a raiz dela está na parte superior e os ramos de diretórios e subdiretórios se estendem abaixo da raiz.



Diretorio / = é o diretório root no topo da hierarquia do sistema de arquivos. O caractere `/` também é usado como separador de diretórios nos nomes de arquivos. Por exemplo, se `etc` for um subdiretório do diretório `/`, chame-o de `/etc`. Do mesmo modo, se o diretório `/etc` contiver um arquivo chamado `issue`, chame o arquivo de `/etc/issue`

Subdiretórios de `/` são usados com fins padronizados para organizar arquivos por tipo e objetivo para facilitar a localização de arquivos. Por exemplo, no diretório root, o subdiretório `/boot` é usado para armazenar os arquivos no boot do sistema.

Como descrever o conteudo do diretorio do sistema de arquivos:

- Conteudo estatico = permance inalterado ate que seja editado ou reconfigurado

- Conteudo dinamico ou variavel = pode ser modficado ou inserido pelos processos ativos
- Conteudo persistente = permanece apos uma reinicilização, como definições de configuração
- Conteudo de tempo de execução de um processo ou do sistema = é excluido na reinicilização

Tabela dos diretorios

Local	Finalidade
/boot	Arquivos para começar o processo de boot.
/dev	Arquivos de dispositivos especiais usados pelo sistema para acessar o hardware.
/etc	Arquivos de configuração específicos do sistema.
/home	O diretório pessoal, onde usuários regulares armazenam seus dados e arquivos de configuração.
/root	Diretório pessoal do superusuário administrativo, `root`.
/run	Dados de tempo de execução de processos iniciados desde o último boot. Esses dados incluem arquivos de ID de processos e arquivos de bloqueio. O conteúdo desse diretório é recriado na reinicialização.
/tmp	Um espaço gravável para arquivos temporários. Arquivos não acessados, alterados nem modificados por 10 dias são excluídos automaticamente desse diretório. O diretório `/var/tmp` também é um diretório temporário, no qual os arquivos que não tiverem sido acessados, alterados ou modificados por mais de 30 dias serão excluídos automaticamente.
/usr	Software instalado, bibliotecas compartilhadas, arquivos incluídos e dados de programas somente leitura. Subdiretórios significativos no diretório /usr incluem os seguintes comandos: /usr/bin: comandos de usuário /usr/sbin: comandos de administração do sistema /usr/local: software personalizado localmente
/var	Dados variáveis específicos do sistema que devem persistir entre boots. Os arquivos que mudam de modo dinâmico; por exemplo, bancos de dados, diretórios de cache, arquivos de log, documentos

	com spool de impressora e conteúdo de sites podem ser encontrados em `/var`.

Especificação de arquivos por nome

Caminhos absolutos e caminhos relativos

O *caminho* de um arquivo ou diretório especifica o local exclusivo no sistema de arquivos. Seguir o caminho de um arquivo atravessa um ou mais subdiretórios nomeados, delimitados por uma barra (/), até chegar ao destino. Os diretórios, também chamados de *pastas*, podem conter outros arquivos e outros subdiretórios. Eles podem ser referenciados da mesma maneira que os arquivos.

#Um caractere de espaço é aceitável no nome de arquivo do Linux. O shell também usa espaços para diferenciar opções e argumentos na linha de comando. Se um comando incluir um arquivo que tenha um espaço no nome, o shell poderá interpretar erroneamente o comando e entender que o nome de arquivo são vários argumentos. Para evitar esse erro, coloque esses nomes de arquivo entre aspas para que o shell interprete o nome como um único argumento.

Caminhos absolutos

Um *caminho absoluto* é um *nome totalmente qualificado*, especificando a localização exata do arquivo na hierarquia do sistema de arquivos.

Ele começa no diretório raiz (`/`) e inclui cada subdiretório que deve ser percorrido para alcançar o arquivo específico. Cada arquivo em um sistema de arquivos tem um único nome de caminho absoluto, que é reconhecido por uma regra simples:

- um nome de arquivo com uma barra (`/`) como primeiro caractere é um nome de caminho absoluto.

Por exemplo, o nome de caminho absoluto do arquivo de log do sistema de mensagens é `/var/log/messages`. Nomes de caminhos absolutos podem ser longos, por isso, os arquivos também podem ser localizados em relação ao diretório de trabalho atual para o prompt do shell.

O diretório de trabalho atual e caminhos relativos

Quando um usuário fizer login e abrir uma janela de comando, normalmente a localização inicial será o diretório pessoal do usuário. Os processos do sistema também têm um diretório inicial. Usuários e processos mudam para outros diretórios conforme necessário. Os termos *diretório de trabalho* ou *diretório de trabalho atual* são referentes ao local atual.

Um caminho *relativo* identifica um único local e especifica somente o caminho necessário para acessar o local no diretório de trabalho. Os nomes de caminhos relativos seguem esta regra:

- um nome de caminho com *qualquer caractere diferente de* uma barra como primeiro caractere é um nome de caminho relativo. Por exemplo, em relação ao diretório `/var`, o arquivo de log de mensagens é `log/messages`.

Navegação pelos caminhos no sistema de arquivos

- O comando `pwd` exibe o nome do caminho completo do diretório de trabalho atual para esse shell. Esse comando ajuda você a determinar a sintaxe para acessar arquivos usando nomes de caminho relativos
- O comando `ls` lista o conteúdo do diretório especificado ou, caso um diretório não seja fornecido, do diretório de trabalho atua
- Use o comando `cd` para alterar o diretório de trabalho atual do shell. Se você não especificar nenhum argumento para o comando, ele será alterado para o diretório pessoal.
- O prompt exibe o caractere til (`~`) quando o diretório de trabalho atual é o diretório pessoal.
- O comando `touch` atualiza o carimbo de data e hora de um arquivo para a data e a hora atuais, sem modificá-lo. Esse comando é útil para a criação de arquivos vazios e pode ser usado para prática, pois o uso desse comando `touch` em um nome de arquivo que não existe faz com que o arquivo seja criado.
- O comando `ls` tem várias opções para a exibição de atributos nos arquivos. As opções mais comuns e `-l` (formato de listagem longa), `-`

`a` (todos os arquivos, incluindo os *ocultos*) e `-R` (recursão, para incluir o conteúdo de todos os subdiretórios).

- No topo da lista há dois diretórios especiais. Um ponto (`.`) se refere ao diretório atual, e dois pontos (`..`) se referem ao diretório pai. Esses diretórios especiais existem em todos os diretórios do sistema e são úteis ao usar comandos de gerenciamento de arquivos.
- Nomes de arquivos começando com um ponto (`.`) indicam arquivos *ocultos*; não é possível visualizá-los na exibição normal com `ls` e outros comando

```
ls -l ~
```

- O comando `cd` tem várias opções. Algumas opções são úteis para praticar cedo e usar com frequência. O comando `cd -` passa para o diretório anterior, no qual o usuário estava *antes* do diretório atual. O exemplo a seguir ilustra esse comportamento, alternando entre dois diretórios, o que é útil ao processar uma série de tarefas semelhantes.
- O comando `cd ..` usa o diretório oculto (`..`)para subir um nível até o diretório pai, sem precisar saber o nome exato desse diretório. O outro diretório oculto (`.`) especifica o *diretório atual* nos comandos em que a localização atual é o argumento de origem ou de destino, evitando a necessidade de digitar o nome de caminho absoluto do diretório.

Gerenciamento de arquivos através de ferramentas de linha de comando

Criação de diretórios

O comando `mkdir` cria um ou mais diretórios ou subdiretórios. Ele considera como um argumento uma lista de caminhos para os diretórios que você deseja criar.

No exemplo a seguir, os arquivos e diretórios são organizados abaixo do diretório `/home/user/Documents`. Use o comando `mkdir` e uma lista de nomes de diretórios separada por espaços para criar vários diretórios.

```
[user@host ~]$cd Documents
[user@host Documents]$mkdir ProjectX ProjectY ProjectZ
[user@host Documents]$ls
ProjectX  ProjectY ProjectZ
```

A comando `mkdir` `-p` (*pai*) cria diretórios pais ausentes para o destino solicitado. No exemplo a seguir, o comando `mkdir` cria três subdiretórios `Chapter` `N` com um comando. A opção `-p` cria o diretório pai `Thesis` ausente.

```
[user@host Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapte
[user@host Documents]$ ls -R Thesis/
Thesis/:
Chapter1  Chapter2  Chapter3

Thesis/Chapter1:

Thesis/Chapter2:

Thesis/Chapter3:
```

Cópia de arquivos e diretórios

O comando `cp` copia um arquivo, criando um arquivo no diretório atual ou em um diretório diferente especificado.

```
[user@host ~]$cd Videos
[user@host Videos]$cp blockbuster1.ogg blockbuster3.ogg
[user@host Videos]$ls -l
total 0
-rw-rw-r--. 1 user user    0 Feb  8 16:23 blockbuster1.ogg
-rw-rw-r--. 1 user user    0 Feb  8 16:24 blockbuster2.ogg
-rw-rw-r--. 1 user user    0 Feb  8 16:34blockbuster3.ogg
```

Você também pode usar o comando `cp` para copiar vários arquivos para um diretório. Nesse cenário, o último argumento deve ser um diretório. Os arquivos copiados mantêm seus nomes originais no novo diretório. Se um arquivo com o

mesmo nome existir no diretório de destino, o arquivo existente será substituído.

No exemplo a seguir, dois diretórios são listados como argumentos, os diretórios `Thesis` e `ProjectX`. O último argumento, o diretório `ProjectX`, é o destino e é válido como um destino. O argumento `Thesis` é ignorado pelo comando `cp`, porque se destina a ser copiado e é um diretório.

```
[user@host Documents]$cp thesis_chapter1.txt thesis_chapter
2.txt Thesis ProjectX
cp: -r not specified; omitting directory 'Thesis'
[user@host Documents]$ls Thesis ProjectX
ProjectX:
thesis_chapter1.txt  thesis_chapter2.txt

Thesis:
Chapter1  Chapter2  Chapter3
```

A cópia do diretório `Thesis` falhou, mas os arquivos `thesis_chapter1.txt` e `thesis_chapter2.txt` foram copiados com êxito.

Você pode copiar diretórios e seus conteúdos usando a opção `-r` do comando `cp`. Lembre-se de que você pode usar os diretórios especiais `.` e `..` em combinações de comandos. No exemplo a seguir, o diretório `Thesis` e seu conteúdo são copiados para o diretório `ProjectY`.

```
[user@host Documents]$cd ProjectY
[user@host ProjectY]$cp -r ../Thesis/ .
[user@host ProjectY]$ls -lR
.:
total 0
drwxr-xr-x. 5 user user 54 Mar  7 15:08 Thesis

./Thesis:
total 0
drwxr-xr-x. 2 user user 6 Mar  7 15:08 Chapter1
drwxr-xr-x. 2 user user 6 Mar  7 15:08 Chapter2
drwxr-xr-x. 2 user user 6 Mar  7 15:08 Chapter3

./Thesis/Chapter1:
```

```
total 0

./Thesis/Chapter2:
total 0

./Thesis/Chapter3:
total 0
```

Transferências de arquivos e diretórios

O comando `mv` move arquivos de um local para outro. Se você pensar no caminho absoluto para um arquivo como seu nome completo, mover um arquivo será efetivamente o mesmo que renomear um arquivo. O conteúdo dos arquivos que são movidos permanece inalterado.

Use o comando `mv` para *renomear* um arquivo. No exemplo a seguir, o comando `mv thesis_chapter2.txt` renomeia o arquivo `thesis_chapter2.txt` para `thesis_chapter2_reviewed.txt` no mesmo diretório.

```
[user@host Documents]$ls -l
-rw-r--r--. 1 user user  7100 Mar  7 14:37 thesis_chapter1.
txt
-rw-r--r--. 1 user user 11431 Mar  7 14:39 thesis_chapter2.
txt
...output omitted...
[user@host Documents]$mv thesis_chapter2.txt thesis_chapter
2_reviewed.txt
[user@host Documents]$ls -l
-rw-r--r--. 1 user user  7100 Mar  7 14:37 thesis_chapter1.
txt
-rw-r--r--. 1 user user 11431 Mar  7 14:39 thesis_chapter2_
reviewed.txt
...output omitted...
```

Remoção de arquivos e diretórios

O comando `rm` remove arquivos. Por padrão, `rm` não remove diretórios. Você pode usar o comando `rm -r` ou a opção `--recursive` para habilitar o

comando `rm` para remover diretórios e seu conteúdo. O comando `rm -r` percorre cada subdiretório primeiro, removendo individualmente seus arquivos antes de remover cada diretório.

Você pode usar a opção `-i` do comando `rm` para solicitar interativamente a confirmação antes da exclusão. Essa opção é essencialmente o oposto de usar a opção `-f` do comando `rm`, que força a remoção sem solicitar a confirmação do usuário.

Você também pode usar o comando `rmdir` para remover diretórios vazios. Use a opção `-r` do comando `rm` para remover diretórios não vazio

```
Exercício orientado: Gerenciamento de arquivos através de fer
Neste exercício, você cria, organiza, copia e remove arquivos

Resultados

Criar, organizar, copiar e remover arquivos e diretórios.

Com o usuário student na máquina workstation, use o comando l

Esse comando prepara seu ambiente e garante que todos os recu

[student@workstation ~]$ lab start files-manage
Instruções

Faça login na máquina servera como o usuário student. No dire

Use o comando ssh para fazer login na máquina servera como o

[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
No diretório pessoal do usuário student, use o comando mkdir




[student@servera ~]$ mkdir Music Pictures Videos
```

```
Use o comando touch para criar conjuntos de arquivos de práti

Crie seis arquivos com nomes nsudoa forma songX.mp3.

Crie seis arquivos com nomes na forma snapX.jpg.

Crie seis arquivos com nomes na forma filmX.avi.

[student@servera ~]$ touch song1.mp3 song2.mp3 song3.mp3 \
song4.mp3 song5.mp3 song6.mp3
[student@servera ~]$ touch snap1.jpg snap2.jpg snap3.jpg \
snap4.jpg snap5.jpg snap6.jpg
[student@servera ~]$ touch film1.avi film2.avi film3.avi \
film4.avi film5.avi film6.avi
[student@servera ~]$ ls -l
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 film1.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film2.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film3.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film4.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film5.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film6.avi
drwxr-xr-x. 2 student student 6 Mar  7 20:58 Music
drwxr-xr-x. 2 student student 6 Mar  7 20:58 Pictures
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap1.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap2.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap3.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap4.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap5.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap6.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 song1.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song2.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song3.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song4.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song5.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song6.mp3
drwxr-xr-x. 2 student student 6 Mar  7 20:58 Videos
Mova os arquivos de música (extensão .mp3) para o diretório M
```

```
[student@servera ~]$ mv song1.mp3 song2.mp3 song3.mp3 \
song4.mp3 song5.mp3 song6.mp3 Music
[student@servera ~]$ mv snap1.jpg snap2.jpg snap3.jpg \
snap4.jpg snap5.jpg snap6.jpg Pictures
[student@servera ~]$ mv film1.avi film2.avi film3.avi \
film4.avi film5.avi film6.avi Videos
[student@servera ~]$ ls -l Music Pictures Videos
Music:
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 song1.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song2.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song3.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song4.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song5.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song6.mp3

Pictures:
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap1.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap2.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap3.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap4.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap5.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap6.jpg

Videos:
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 film1.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film2.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film3.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film4.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film5.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film6.avi
```

Crie três subdiretórios para organizar seus arquivos e nomeie friends, family e work. Use um único comando para criar todos ao mesmo tempo.

```
[student@servera ~]$ mkdir friends family work
[student@servera ~]$ ls -l
total 0
drwxr-xr-x. 2 student student   6 Mar  7 21:01 family
drwxr-xr-x. 2 student student   6 Mar  7 21:01 friends
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Music
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Pictures
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Videos
drwxr-xr-x. 2 student student   6 Mar  7 21:01 work

Copie arquivos que contenham números 1 e 2 para o diretório f
contenham números 3 e 4 para o diretório family. Tenha em men
cópias, portanto, os arquivos originais permanecerão em seus
concluir a etapa.

Ao copiar arquivos de vários locais para um único local, a Re

Copie os arquivos que contêm números 1 e 2 para o diretório f

[student@servera ~]$ cd friends
[student@servera friends]$ cp ~/Music/song1.mp3 ~/Music/song2
~/Pictures/snap1.jpg ~/Pictures/snap2.jpg ~/Videos/film1.avi
~/Videos/film2.avi .
[student@servera friends]$ ls -l
total 0
-rw-r--r--. 1 student student 0 Mar  7 21:02 film1.avi
-rw-r--r--. 1 student student 0 Mar  7 21:02 film2.avi
-rw-r--r--. 1 student student 0 Mar  7 21:02 snap1.jpg
-rw-r--r--. 1 student student 0 Mar  7 21:02 snap2.jpg
-rw-r--r--. 1 student student 0 Mar  7 21:02 song1.mp3
-rw-r--r--. 1 student student 0 Mar  7 21:02 song2.mp3
Copie os arquivos que contêm números 3 e 4 para o diretório f

[student@servera friends]$ cd ../family
[student@servera family]$ cp ~/Music/song3.mp3 ~/Music/song4.
~/Pictures/snap3.jpg ~/Pictures/snap4.jpg ~/Videos/film3.avi
~/Videos/film4.avi .
```

```
[student@servera family]$ ls -l
total 0
total 0
-rw-r--r--. 1 student student 0 Mar  7 21:04 film3.avi
-rw-r--r--. 1 student student 0 Mar  7 21:04 film4.avi
-rw-r--r--. 1 student student 0 Mar  7 21:04 snap3.jpg
-rw-r--r--. 1 student student 0 Mar  7 21:04 snap4.jpg
-rw-r--r--. 1 student student 0 Mar  7 21:04 song3.mp3
-rw-r--r--. 1 student student 0 Mar  7 21:04 song4.mp3
Copie os diretórios family e friends e seus conteúdos para o
```

```
[student@servera family]$ cd ../work
[student@servera work]$ cp -r ~/family ~/friends .
[student@servera work]$ ls -l
total 0
drwxr-xr-x. 2 student student 108 Mar  7 21:05 family
drwxr-xr-x. 2 student student 108 Mar  7 21:05 friends
```

```
As tarefas do seu projeto agora estão completas, e é hora de
comando rm -r para excluir recursivamente os subdiretórios fa
conteúdos.
```

```
[student@servera work]$ cd ..
[student@servera ~]$ rm -r family friends work
[student@servera ~]$ ls -l
total 0
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Music
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Pictures
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Videos
Retorne ao sistema workstation como o usuário student.
```

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
Encerramento
```

```
Na máquina workstation, altere para o diretório pessoal do us
```

```
[student@workstation ~]$ lab finish files-manage
Isso conclui a seção.
```

Criação de links entre arquivos

Gerenciamento de links entre arquivos

É possível criar vários nomes que apontam para o mesmo arquivo. Esses nomes de arquivo são chamados de *links*.

Você pode criar dois tipos de links: um *link físico* ou um *link simbólico* (às vezes chamado de *link soft*). Cada um tem suas vantagens e desvantagens.

Criação de links físicos

Todo arquivo inicia com um único link físico, desde seu nome inicial até os dados no sistema de arquivos. Quando você cria um link físico para um arquivo, cria outro nome que aponta para os mesmos dados. O novo link físico age exatamente como o nome do arquivo original. Depois que o link é criado, você não verá diferença entre o novo link físico e o nome original do arquivo.

Você pode determinar se um arquivo tem vários links físicos usando o comando `ls -l`. Um item que ele relata é a *contagem de links* de cada arquivo, o número de links físicos que o arquivo possui. No próximo exemplo, a contagem de links do arquivo `newfile.txt` é 1. Ele tem exatamente um caminho absoluto, que é o local `/home/user/newfile.txt` .

```
[user@host ~]$pwd
/home/user
[user@host ~]$ls -l newfile.txt
-rw-r--r--.1 user user 0 Mar 11 19:19 newfile.txt
```

Criar link fisico:

Você pode usar o comando `ln` para criar um link físico (outro nome de arquivo) que aponte para um arquivo existente. O comando precisa de pelo menos dois argumentos, um caminho para o arquivo existente e o caminho para o link

físico que você deseja criar.

O exemplo a seguir cria um link físico chamado `newfile-hlink2.txt` para o arquivo existente `newfile.txt` no diretório `/tmp`.

```
[user@host ~]$ln newfile.txt /tmp/newfile-hlink2.txt
[user@host ~]$ls -l newfile.txt /tmp/newfile-hlink2.txt
-rw-rw-r--.2 user user 12 Mar 11 19:19 newfile.txt
-rw-rw-r--.2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.
txt
```

Para determinar se dois arquivos estão vinculados, use a opção `-i` do comando `ls` para listar o *número de inode* de cada arquivo. Se os arquivos estiverem no mesmo sistema de arquivos e seus números de inode forem os mesmos, os arquivos são links físicos apontando para o conteúdo de arquivo dos mesmos dados.

- **Importante:**
  - Os links físicos que fazem referência ao mesmo arquivo compartilham a estrutura inode com a contagem de links, as permissões de acesso, a propriedade de usuário e grupo, os carimbos de data e hora e o conteúdo de arquivo. Quando essas informações são alteradas para um link físico, os outros links físicos para o mesmo arquivo também mostram as novas informações. Esse comportamento ocorre porque cada link físico aponta para os mesmos dados no dispositivo de armazenamento.

Mesmo que o arquivo original seja excluído, o conteúdo do arquivo ainda estará disponível, desde que pelo menos um link físico exista. Os dados são excluídos do armazenamento somente quando o último link físico é excluído, o que torna o conteúdo do arquivo não referenciado por qualquer link físico.

Limitações de links físicos

Em primeiro lugar, os links físicos só podem ser usados com arquivos regulares. Você não pode usar o comando `ln` para criar um link físico para um

diretório ou arquivo especial.

Em segundo lugar, os links físicos só podem ser usados se ambos os arquivos estiverem no mesmo *sistema de arquivos*. A hierarquia do sistema de arquivos pode ser composta de vários dispositivos de armazenamento. Dependendo da configuração do sistema, quando você mudar para um novo diretório, esse diretório e seu conteúdo poderão ser armazenados em um sistema de arquivos diferente.

Você pode usar o comando `df` para listar os diretórios que estão em sistemas de arquivos diferentes. Por exemplo, você pode ver a saída desta maneira:

```
[user@host ~]$df
Filesystem                   1K-blocks    Used Available Us
e% Mounted on
devtmpfs                        886788       0    886788
0% /dev
tmpfs                           902108       0    902108
0% /dev/shm
tmpfs                           902108    8696    893412
1% /run
tmpfs                           902108       0    902108
0% /sys/fs/cgroup
/dev/mapper/rhel_rhel9--root  10258432 1630460   8627972  1
6% /
/dev/sda1                      1038336  167128    871208  1
7% /boot
tmpfs                           180420       0    180420
0% /run/user/1000
```

#Arquivos em dois diretórios "montados em" diferentes e seus subdiretórios estão em sistemas de arquivos diferentes. Assim, no sistema deste exemplo, você pode criar um link físico entre os arquivos `/var/tmp/link1` e `/home/user/file` porque ambos são subdiretórios do diretório `/`, mas não de qualquer outro diretório na lista. No entanto, não é possível criar um link físico entre os arquivos `/boot/test/badlink` e `/home/user/file`. O primeiro arquivo está em um subdiretório do diretório `/boot` (na lista "montado em") e está no sistema de arquivos `/dev/sda1`. O segundo arquivo está no sistema de arquivos `/dev/mapper/rhel_rhel9–root`.

Criação de links simbólicos

A opção `-s` do comando `ln` cria um link simbólico, também chamado de "link soft". Um link simbólico não é um arquivo normal, mas um tipo especial de arquivo que aponta para outro arquivo ou diretório existente.

Os links simbólicos têm algumas vantagens sobre links físicos:

- Os links simbólicos podem vincular dois arquivos em diferentes sistemas de arquivos.
- Os links simbólicos podem apontar para um diretório ou arquivo especial, não apenas um arquivo comum

No exemplo a seguir, o comando `ln -s` cria um link simbólico para o arquivo `/home/user/newfile-link2.txt`. O nome do link simbólico é `/tmp/newfile-symlink.txt`

```
[user@host ~]$ln -s /home/user/newfile-link2.txt /tmp/newfi
le-symlink.txt
[user@host ~]$ls -l newfile-link2.txt /tmp/newfile-symlink.
txt
-rw-rw-r--.1 user user 12 Mar 11 19:19 newfile-link2.txt
lrwxrwxrwx.1 user user 11 Mar 11 20:59 /tmp/newfile-symlin
k.txt -> /home/user/newfile-link2.txt
[user@host ~]$cat /tmp/newfile-symlink.txt
Symbolic Hello World
```

No exemplo anterior, o primeiro caractere da listagem longa para `/tmp/newfile-symlink.txt` é `l` (letra l), em vez de `-`. Esse caractere indica que o arquivo é um link simbólico e não um arquivo normal.

Quando o arquivo regular original é excluído, o link simbólico continua apontando para o arquivo, mas o destino some. Um link simbólico que aponta para um arquivo ausente é chamado de "link simbólico pendente".

```
[user@host ~]$rm -f newfile-link2.txt
[user@host ~]$ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx.1 user user 11 Mar 11 20:59 /tmp/newfile-symlin
k.txt -> /home/user/newfile-link2.txt
```

```
[user@host ~]$cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```

Importante:

- Um efeito colateral do link simbólico pendente no exemplo anterior é que, se você criar posteriormente um novo arquivo com o mesmo nome do arquivo excluído (`/home/user/newfile-link2.txt`), o link não estará mais "pendurado" e apontará para o novo arquivo. Os links físicos não funcionam assim. Se você excluir um link físico e, em seguida, usar ferramentas normais (ao invés de `ln`) para criar um arquivo com o mesmo nome, o novo arquivo não será vinculado ao arquivo antigo. Considere a seguinte maneira de comparar links físicos e links simbólicos, para entender como eles funcionam:
  - Um link físico aponta um nome para dados em um dispositivo de armazenamento
  - Um link simbólico aponta um nome para outro nome, que aponta para dados em um dispositivo de armazenamento

Um link simbólico pode apontar para um diretório. Nesse caso, o link simbólico atuará como o diretório.

Se você usar `cd` para alterar para o link simbólico, o diretório de trabalho atual se tornará o diretório vinculado.

Algumas ferramentas podem acompanhar o fato de você ter seguido um link simbólico para chegar lá. Por exemplo, por padrão `cd` atualiza seu diretório de trabalho atual usando o nome do link simbólico, em vez do nome do diretório real.

Se você quiser atualizar o diretório de trabalho atual usando o nome do diretório real, poderá usar a opção `-P` .

O exemplo a seguir cria um link simbólico denominado `/home/user/configfiles` que aponta para o diretório `/etc` .

```
[user@host ~]$ln -s /etc /home/user/configfiles
[user@host ~]$cd /home/user/configfiles
[user@host configfiles]$pwd
/home/user/configfiles
```

```
[user@host configfiles]$cd -P /home/user/configfiles
[user@host etc]$pwd
/etc
```

```
Exercício orientado: Criação de links entre arquivos
Neste exercício, você cria links físicos e links simbólicos e

Resultados

Criar links físicos e links simbólicos entre arquivos.

Com o usuário student na máquina workstation, use o comando la
para este exercício.

Esse comando prepara seu ambiente e garante que todos os recu
disponíveis.

[student@workstation ~]$ lab start files-make
Instruções

Use o comando ssh para fazer login na máquina servera como o

[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$

Crie um link físico chamado /home/student/links/file.hardlink
/home/student/files/target.file. Verifique a contagem de link
o novo arquivo vinculado.

Veja a contagem de links para o arquivo /home/student/files/t

[student@servera ~]$ ls -l files/target.file
total 4
-rw-r--r--. 1 student student 11 Mar  3 06:51 files/target.fi

Crie um link físico chamado /home/student/links/file.hardlink
/home/student/files/target.file.
```

```
[student@servera ~]$ ln /home/student/files/target.file \
/home/student/links/file.hardlink
Verifique a contagem de links para o arquivo /home/student/fi

[student@servera ~]$ ls -l files/target.file links/file.hardl
-rw-r--r--. 2 student student 11 Mar  3 06:51 files/target.fi
-rw-r--r--. 2 student student 11 Mar  3 06:51 links/file.hard
Crie um link simbólico denominado /home/student/tempdir que a

Crie um link simbólico chamado /home/student/tempdir e vincul

[student@servera ~]$ ln -s /tmp /home/student/tempdir
Use o comando ls -l para verificar o link simbólico criado re

[student@servera ~]$ ls -l /home/student/tempdir
lrwxrwxrwx. 1 student student 4 Mar  3 06:55 /home/student/te
Retorne ao sistema workstation como o usuário student.

[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
Encerramento

Na máquina workstation, altere para o diretório pessoal do us

[student@workstation ~]$ lab finish files-make
```

Correspondência de nomes de arquivos com expansões de shell

Objetivos

Executar com eficiência comandos que afetam muitos arquivos usando recursos de correspondência de padrões do shell Bash.

Expansões de linha de comando

Quando digitamos no prompt do shell, o shell processa essa linha de comando por meio de várias *expansões* antes de executá-la

Você pode usar essas expansões de shell para realizar tarefas complexas que, de outra forma, seriam difíceis ou impossíveis.

Principais expansões que o shell do Bash executa:

- *Expansão de chave*, que pode gerar várias cadeias de caracteres

- *Expansão de til*, que expande para um caminho para um diretório pessoal de um usuário

- *Expansão de variável*, que substitui o texto pelo valor armazenado em uma variável do shell

- *Substituição de comando*, que substitui texto pela saída de um comando

- *Expansão de nome de caminho*, que ajuda a selecionar um ou mais arquivos por correspondência de padrões

A expansão de nome de caminho,

anteriormente chamada de **GLOBBING**, é um dos recursos mais úteis do Bash.

Pois pode gerenciar muitor arquivos. Ao usar ***metacaracteres*** que se **"expandem"** para corresponder a **nomes de caminho** e de arquivo que são procurados, os comandos são executados em um conjunto específico de arquivos de uma só vez.

Expansão de nome de caminho e correspondência de padrões

A expansão de nome de caminho expande um padrão de caracteres especiais que representam curingas ou classes de caracteres em uma lista de nomes de arquivos que correspondem ao padrão.

Antes de executar seu comando, o shell substitui o padrão pela lista de nomes de arquivo correspondentes. Se o padrão não corresponder a nada, o shell tentará usar o padrão como um argumento literal para o comando que executa. A tabela a seguir lista **metacaracteres e classes de padrões** comuns usados para correspondência de padrões.

Tabela 3.2. Tabela de metacaracteres e correspondências

Padrão	Correspondencias
*	Qualquer string com zero ou mais caracteres
?	Qualquer caracter unico
[abc...]	Qualquer caractere na classe entre colchetes
[!*abc...*]	Qualquer caractere que *não* esteja na classe entre colchetes
[^*abc...*]	Qualquer caractere que *não* esteja na classe entre colchetes
[[:alpha:]]	Qualquer caractere alfabético
[[:lower:]]	Qualquer caractere em minúsculas
[[:upper:]]	Qualquer caractere em maiusculo
[[:alnum:]]	Qualquer caractere alfabetico ou numerico
[[:punct:]]	Qualquer caractere imprimível que não seja alfanumérico nem um espaço
[[:digit:]]	Qualquer dígito único de 0 a 9
[[:space:]]	Qualquer caractere de espaço em branco único, o que pode incluir tabulações, novas linhas, retornos de carro, avanços de página ou espaços

Para o próximo exemplo, digamos que você executou os comandos a seguir para criar alguns arquivos de amostra.

```
[user@host ~]$mkdir glob; cd glob
[user@host glob]$touch alfa bravo charlie delta echo able b
```

```
aker cast   dog easy
[user@host glob]$ls
able  alfa  baker  bravo  cast  charlie  delta  dog  easy
echo
[user@host glob]$
```

No próximo exemplo, os dois primeiros comandos usam correspondências de padrão simples com o asterisco (*),

Para corresponder a todos os nomes de arquivos que começam com "a" e todos os nomes de arquivos que contêm um "a", respectivamente.

O terceiro comando usa o asterisco e colchetes para corresponder a todos os nomes de arquivos que começam com "a" ou "c".

```
[user@host glob]$ls a*
able  alfa
[user@host glob]$ls *a*
able  alfa  baker  bravo  cast  charlie  delta  easy
[user@host glob]$ls [ac]*
able  alfa  cast  charlie
```

`O próximo exemplo também usa caracteres de ponto de interrogação (?) para corresponder a alguns desses nomes de arquivo. Os dois comandos correspondem apenas a nomes de arquivos com quatro e cinco caracteres, respectivamente.`

```
[user@host glob]$ls ????
able  cast  easy  echo
[user@host glob]$ls ?????
baker  bravo  delta
```

Expansão de chave

- A expansão de chave é usada para gerar strings de caracteres distintas.
- As chaves contêm uma lista de strings separadas por vírgula ou uma expressão de sequência.
- O resultado inclui o texto anterior ou o posterior à definição de chave

- As expansões de chave podem ser aninhadas uma dentro da outra
- Você também pode usar a sintaxe de ponto duplo (..), que se expande para uma sequência. Por exemplo, a sintaxe de ponto duplo `{m..p}` entre chaves se expande para `m n o p`.

```
[user@host glob]$echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[user@host glob]$echo file{1..3}.txt
file1.txt file2.txt file3.txt
[user@host glob]$echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[user@host glob]$echo file{a,b}{1,2}.txt
filea1.txt filea2.txt fileb1.txt fileb2.txt
[user@host glob]$echo file{a{1,2},b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
```

- Um uso prático da expansão de chaves é criar vários arquivos ou diretórios.

```
[user@host glob]$mkdir ../RHEL{7,8,9}
[user@host glob]$ls ../RHEL*
RHEL7 RHEL8 RHEL9
```

Expansão de til

- O caractere til (~) corresponde ao diretório pessoal do usuário atual.
- Se uma string diferente de uma barra (/) for iniciada, o shell interpretará a string até essa barra como um nome de usuário,
- se houver uma correspondência, e substituirá a string pelo caminho absoluto para o diretório pessoal desse usuário. Se nenhum nome de usuário for correspondente, o shell usará um til real seguido da string.
- `No exemplo a seguir, o comando echo é usado para exibir o valor do caractere til.`

```
[user@host glob]$echo ~root
/root
[user@host glob]$echo ~user
```

```
/home/user
[user@host glob]$echo ~/glob
/home/user/glob
[user@host glob]$echo ~nonexistinguser
~nonexistinguser
```

Expansão variável

- Uma variável age como um contêiner nomeado que armazena um valor na memória.
- As variáveis simplificam o acesso e a modificação dos dados armazenados a partir da linha de comando ou dentro de um script de shell.
- Você pode atribuir dados como um valor a uma variável com a seguinte sintaxe:

```
[user@host ~]$VARIABLENAME=value
```

- Você pode usar a expansão variável para converter o nome da variável para o valor na linha de comando. Se uma string começa com um símbolo de dólar ($), o shell tenta usar o restante dessa string como um nome de variável e substituir pelo valor da variável.

```
[user@host ~]$USERNAME=operator
[user@host ~]$echo $USERNAME
operator
```

Para prevenir erros devido a outras expansões de shell, você pode colocar o nome da variável entre chaves, por exemplo `${VARIABLENAME}`

Os nomes de variáveis podem conter apenas letras (maiúsculas e minúsculas), números e sublinhados. Os nomes das variáveis diferenciam maiúsculas de minúsculas e não podem começar com um número.

Substituição de comandos

A substituição de comandos habilita a saída de um comando para substituir o próprio comando na linha de comando.

A substituição de comandos ocorre quando um comando é colocado entre parênteses e precedido por um símbolo de dólar (`$`). A forma `$( command )` pode aninhar várias expansões de comandos, uma dentro da outra.

```
[user@host glob]$echo Today is $(date +%A).
Today is Wednesday.
[user@host glob]$echo The time is $(date +%M) minutes past
$(date +%l%p).
The time is 26 minutes past 11AM.
```

Proteção de argumentos contra expansão

- Vários caracteres têm um significado especial no shell Bash.
- Para impedir expansões de shell em partes de sua linha de comando, você pode usar aspas e escapes em caracteres e strings.
- A barra invertida (`\`) é um caractere de escape no shell Bash. Ela protege o caractere seguinte contra expansão.

```
[user@host glob]$echo The value of $HOME is your home direc
tory.
The value of /home/user is your home directory.
[user@host glob]$echo The value of \$HOME is your home dire
ctory.
The value of $HOME is your home directory.
```

- No exemplo anterior, com o símbolo de dólar protegido contra expansão, o Bash o trata como um caractere regular, sem expansão de variável em `$HOME`.
- **Para proteger strings mais longas, você pode usar aspas simples (`'`) ou duplas (`"`) para delimitar strings.]**
    - Elas têm efeitos ligeiramente diferentes
        - As aspas simples param toda a expansão do shell

    - As aspas duplas param a *maior parte* da expansão do shell.

- Aspas duplas impedem que caracteres especiais, com exceção:
    - do cifrão (`$`),
    - da barra invertida (`\`),
    - do acento grave (`),
    - e do ponto de exclamação (`!`),

Operem dentro do texto entre aspas. Aspas duplas bloqueiam a expansão de nome de caminho, mas ainda permitem que a substituição de comando e a expansão de variável ocorram.

```
[user@host glob]$myhost=$(hostname -s); echo $myhost
host
[user@host glob]$echo "***** hostname is ${myhost} *****"
***** hostname is host *****
```

`Use aspas simples para interpretar` *`todo`* `o text literalmente.`

Exercicio

```
Laboratório Aberto: Gerenciar arquivos a partir da linha de c
Neste laboratório, você cria, move e remove arquivos e diretó
eficiente usando o shell e várias técnicas de correspondência

Resultados

Usar curingas para localizar e manipular arquivos.

Com o usuário student na máquina workstation, use o comando l
seu sistema para este exercício.

Esse comando prepara seu ambiente e garante que todos os recu
```

```
estejam disponíveis.

[student@workstation ~]$ lab start files-review
Instruções

Use o comando ssh para fazer login na máquina serverb como o
A configuração do sistema é compatível com o uso de chaves SS

[student@workstation ~]$ ssh student@serverb
...output omitted...

Crie um diretório chamado project_plans no diretório Document
é colocado no diretório pessoal do usuário student. Crie dois
diretório project_plans chamados season1_project_plan.odf e s

Dica: se o diretório ~/Documents não existir, use a opção -p
para criá-lo.

[student@serverb ~]$ mkdir -p ~/Documents/project_plans
[student@serverb ~]$ touch \
~/Documents/project_plans/{season1,season2}_project_plan.odf
[student@serverb ~]$ ls -lR Documents/
Documents/:
total 0
drwxr-xr-x. 2 student student 70 Mar  7 03:50 project_plans

Documents/project_plans:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:50 season1_project_
-rw-r--r--. 1 student student 0 Mar  7 03:50 season2_project_
Crie conjuntos de arquivos de prática vazios para usar neste
reconhecer imediatamente o atalho da expansão do shell, use a
praticar. Use o preenchimento com Tab do shell para localizar
arquivo. Crie 12 arquivos com nomes tv_seasonX_episodeY.ogg n
Substitua X pelo número da temporada e Y pelo episódio da tem
de seis episódios cada.

[student@serverb ~]$ touch tv_season{1..2}_episode{1..6}.ogg
```

```
[student@serverb ~]$ ls tv*
tv_season1_episode1.ogg  tv_season1_episode5.ogg  tv_season2_
tv_season1_episode2.ogg  tv_season1_episode6.ogg  tv_season2_
tv_season1_episode3.ogg  tv_season2_episode1.ogg  tv_season2_
tv_season1_episode4.ogg  tv_season2_episode2.ogg  tv_season2_
Como autor de uma série de sucesso de romances de mistério, o
seu próximo best-seller estão sendo editados para publicação.
os nomes mystery_chapterX.odf. Substitua X pelos números 1 a

[student@serverb ~]$ touch mystery_chapter{1..8}.odf
[student@serverb ~]$ ls mys*
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.
mystery_chapter3.odf  mystery_chapter6.odf
Use um comando único para criar dois subdiretórios chamados s

Crie dois subdiretórios chamados season1 e season2 no diretór

[student@serverb ~]$ mkdir -p Videos/season{1..2}
[student@serverb ~]$ ls Videos
season1  season2
Mova os episódios de TV adequados para os subdiretórios da te

[student@serverb ~]$ mv tv_season1* Videos/season1
[student@serverb ~]$ mv tv_season2* Videos/season2
[student@serverb ~]$ ls -R Videos
Videos:
season1  season2

Videos/season1:
tv_season1_episode1.ogg  tv_season1_episode3.ogg  tv_season1_
tv_season1_episode2.ogg  tv_season1_episode4.ogg  tv_season1_

Videos/season2:
tv_season2_episode1.ogg  tv_season2_episode3.ogg  tv_season2_
tv_season2_episode2.ogg  tv_season2_episode4.ogg  tv_season2_
Crie uma hierarquia de diretório em dois níveis com um comand
```

```
Crie o diretório my_bestseller no diretório Documents. Crie o

[student@serverb ~]$ mkdir -p Documents/my_bestseller/chapter
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller  project_plans

Documents/my_bestseller:
chapters

Documents/my_bestseller/chapters:

Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
Crie três subdiretórios chamados editor, changes e vacation n

[student@serverb ~]$ mkdir Documents/my_bestseller/{editor,ch
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller  project_plans

Documents/my_bestseller:
changes  chapters  editor  vacation

Documents/my_bestseller/changes:

Documents/my_bestseller/chapters:

Documents/my_bestseller/editor:

Documents/my_bestseller/vacation:

Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
Change to the chapters directory. Use o atalho do diretório p

Envie os dois primeiros capítulos ao editor para revisão. Mov
```

```
Durante as férias, você pretende escrever os capítulos 7 e 8.

Altere para o diretório chapters e use o atalho do diretório

[student@serverb ~]$ cd Documents/my_bestseller/chapters
[student@serverb chapters]$ mv ~/mystery_chapter* .
[student@serverb chapters]$ ls
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.
mystery_chapter3.odf  mystery_chapter6.odf
Mova os dois primeiros capítulos para o diretório editor. Use

[student@serverb chapters]$ mv mystery_chapter{1..2}.odf ../e
[student@serverb chapters]$ ls
mystery_chapter3.odf  mystery_chapter5.odf  mystery_chapter7.
mystery_chapter4.odf  mystery_chapter6.odf  mystery_chapter8.
[student@serverb chapters]$ ls ../editor
mystery_chapter1.odf  mystery_chapter2.odf
Use um comando único para mover os capítulos 7 e 8 do diretór

[student@serverb chapters]$ mv mystery_chapter{7,8}.odf ../va
[student@serverb chapters]$ ls
mystery_chapter3.odf  mystery_chapter5.odf
mystery_chapter4.odf  mystery_chapter6.odf
[student@serverb chapters]$ ls ../vacation
mystery_chapter7.odf  mystery_chapter8.odf
Mude seu diretório de trabalho para ~/Videos/season2 e copie

Mude seu diretório de trabalho para ~/Videos/season2 e copie

[student@serverb chapters]$ cd ~/Videos/season2
[student@serverb season2]$ cp *episode1.ogg ~/Documents/my_be
Use um único comando cd para mudar de seu diretório de trabal

[student@serverb season2]$ cd ~/Documents/my_bestseller/vacat
[student@serverb vacation]$ ls
mystery_chapter7.odf  mystery_chapter8.odf  tv_season2_episod
[student@serverb vacation]$ cd -
```

```
/home/student/Videos/season2
[student@serverb season2]$ cp *episode2.ogg ~/Documents/my_be
[student@serverb season2]$ cd -
/home/student/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf  tv_season2_episode1.ogg
mystery_chapter8.odf  tv_season2_episode2.ogg
Os autores dos capítulos 5 e 6 querem testar possíveis mudanç

[student@serverb vacation]$ cd ~/Documents/my_bestseller
[student@serverb my_bestseller]$ cp chapters/mystery_chapter[
[student@serverb my_bestseller]$ ls chapters
mystery_chapter3.odf  mystery_chapter5.odf
mystery_chapter4.odf  mystery_chapter6.odf
[student@serverb my_bestseller]$ ls changes
mystery_chapter5.odf  mystery_chapter6.odf
Altere o diretório atual para o diretório changes e use o com

Ao usar a substituição de comandos com o comando date +%s, fa

[student@serverb my_bestseller]$ cd changes
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%F).odf
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%s).odf
[student@serverb changes]$ ls
mystery_chapter5_1646644424.odf  mystery_chapter5.odf
mystery_chapter5_2022-03-07.odf  mystery_chapter6.odf
Após uma nova revisão, você decide que as alterações no gráfi

Se necessário, navegue até o diretório changes e exclua todos

Altere para o diretório pai do diretório changes. Tente exclu

Quando as férias terminarem, o diretório vacation não será ma

Quando tiver concluído, retorne ao diretório pessoal do usuár
```

```
Exclua o diretório changes. Mude para o diretório pai do dire

[student@serverb changes]$ rm mystery*
[student@serverb changes]$ cd ..
[student@serverb my_bestseller]$ rm changes
rm: cannot remove 'changes': Is a directory
[student@serverb my_bestseller]$ rmdir changes
[student@serverb my_bestseller]$ ls
chapters  editor  vacation
Exclua o diretório vacation usando o comando rm com a opção -

[student@serverb my_bestseller]$ rm -r vacation
[student@serverb my_bestseller]$ ls
chapters  editor
[student@serverb my_bestseller]$ cd
[student@serverb ~]$
Crie um link físico para o arquivo ~/Documents/project_plans/

Dica: se o diretório ~/Documents/backups não existir, use o c

Crie um link físico para o arquivo ~/Documents/project_plans/

[student@serverb ~]$ mkdir ~/Documents/backups
[student@serverb ~]$ ln ~/Documents/project_plans/season2_pro
~/Documents/backups/season2_project_plan.odf.back
[student@serverb ~]$ ls -lR ~/Documents/
/home/student/Documents/:
total 0
drwxr-xr-x. 2 student student 43 Mar  7 04:18 backups
drwxr-xr-x. 4 student student 36 Mar  7 04:16 my_bestseller
drwxr-xr-x. 2 student student 70 Mar  7 03:50 project_plans

/home/student/Documents/backups:
total 0
-rw-r--r--. 2 student student 0 Mar  7 03:50 season2_project_

/home/student/Documents/my_bestseller:
total 0
```

```
drwxr-xr-x. 2 student student 118 Mar  7 04:07 chapters
drwxr-xr-x. 2 student student  62 Mar  7 04:06 editor

/home/student/Documents/my_bestseller/chapters:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter3
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter4
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter5
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter6

/home/student/Documents/my_bestseller/editor:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter1
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter2

/home/student/Documents/project_plans:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:50 season1_project_
-rw-r--r--. 2 student student 0 Mar  7 03:50 season2_project_
Observe que a contagem de links é 2 para ambos os arquivos se

Retorne ao sistema workstation como o usuário student.

[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
Avaliação

Com o usuário student na máquina workstation, use o comando l

[student@workstation ~]$ lab grade files-review
Encerramento

Na máquina workstation, altere para o diretório pessoal do us

[student@workstation ~]$ lab finish files-review
Isso conclui a seção.
```

Sumário

- Os arquivos em um sistema Linux são organizados em uma única árvore de diretório invertida, conhecida como hierarquia do sistema de arquivos.
- Os caminhos absolutos começam com um caractere de barra (`/`) e especificam a localização de um arquivo na hierarquia do sistema de arquivos.
- Caminhos relativos não começam com um caractere de barra.
- Caminhos relativos especificam uma localização de arquivo em relação ao diretório de trabalho atual.
- Você pode usar comandos em combinação com os caracteres especiais ponto (`.`), ponto duplo (`..`) e til (`~`) para se referir a um local de arquivo no sistema de arquivos.
- Os comandos `mkdir`, `rmdir`, `cp`, `mv` e `rm` são os principais comandos para gerenciar arquivos no Linux.
- Os links físicos e os links simbólicos são maneiras diferentes de ter vários nomes de arquivos apontando para os mesmos dados.
- O shell Bash fornece recursos de correspondência, expansão e substituição de padrões para ajudar você a executar comandos de maneira eficiente.