



## 103.3 Gerenciamento básico de arquivos

### Referência ao LPI objective

LPIC-1 v5, Exam 101, Objective 103.3

### Peso

4

### Áreas chave de conhecimento

- Copiar, mover e remover arquivos e diretórios individualmente.
- Copiar múltiplos arquivos e diretórios recursivamente.
- Remover arquivos e diretórios recursivamente.
- Uso simples e avançado dos caracteres curinga nos comandos.
- Usar o comando `find` para localizar e tratar arquivos tomando como base o tipo, o tamanho ou a data.
- Uso dos utilitários `tar`, `cpio` e `dd`.

### Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `cp`
- `find`
- `mkdir`
- `mv`
- `ls`
- `rm`
- `rmdir`

- `touch`
- `tar`
- `cpio`
- `dd`
- `file`
- `gzip`
- `gunzip`
- `bzip2`
- `bunzip2`
- File globbing (englobamento de arquivos)



## 103.3 Lição 1

<b>Certificação:</b>	LPIC-1
<b>Versão:</b>	5.0
<b>Tópico:</b>	103 Comandos GNU e Unix
<b>Objetivo:</b>	103.3 Gerenciamento básico de arquivos
<b>Lição:</b>	1 de 2

### Introdução

No Linux, tudo são arquivos, então é importantíssimo saber como manipulá-los. Nesta lição, trataremos das operações básicas em arquivos.

Em geral, um usuário Linux precisa saber navegar pelo sistema de arquivos, copiar arquivos de um local para outro e excluir arquivos. Também falaremos dos comandos associados ao gerenciamento de arquivos.

Um arquivo é uma entidade que armazena dados e programas. Consiste em conteúdo e metadados (tamanho do arquivo, proprietário, data de criação, permissões). Os arquivos são organizados em diretórios. Um diretório é um arquivo que armazena outros arquivos.

Dentre os diferentes tipos de arquivos, temos:

#### Arquivos regulares

armazenam dados e programas.

## Diretórios

contêm outros arquivos.

## Arquivos especiais

usados para entrada e saída de dados.

Claro, existem outros tipos de arquivos, mas eles estão além do escopo desta lição. Mais tarde, ensinaremos como identificar esses outros tipos.

## Manipulação de arquivos

### Usando `ls` para listar arquivos

O comando `ls` é uma das ferramentas de linha de comando mais importantes que precisamos aprender para navegar no sistema de arquivos.

Em sua forma mais básica, o `ls` lista *somente* nomes de arquivos e diretórios:

```
$ ls
Desktop Downloads  emp_salary file1 Music Public Videos
Documents emp_name  examples.desktop file2 Pictures Templates
```

Quando usado com `-l`, conhecido como formato de “listagem longa”, ele mostra as permissões de arquivo ou diretório, proprietário, tamanho, data de modificação, hora e nome:

```
$ ls -l
total 60
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Desktop
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Documents
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Downloads
-rw-r--r-- 1 frank frank 21 Sep 7 12:59 emp_name
-rw-r--r-- 1 frank frank 20 Sep 7 13:03 emp_salary
-rw-r--r-- 1 frank frank 8980 Apr 1 2018 examples.desktop
-rw-r--r-- 1 frank frank 10 Sep 1 2018 file1
-rw-r--r-- 1 frank frank 10 Sep 1 2018 file2
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Music
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Pictures
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Public
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Templates
drwxr-xr-x 2 frank frank 4096 Apr 1 2018 Videos
```

O primeiro caractere na saída indica o tipo de arquivo:

-

para um arquivo regular.

d

para um diretório.

c

para um arquivo especial.

Para mostrar o tamanho dos arquivos em um formato legível por humanos, adicione a opção `-h`:

```
$ ls -lh
total 60K
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Desktop
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Documents
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Downloads
-rw-r--r--  1 frank frank  21 Sep 7 12:59 emp_name
-rw-r--r--  1 frank frank  20 Sep 7 13:03 emp_salary
-rw-r--r--  1 frank frank 8.8K Apr 1 2018 examples.desktop
-rw-r--r--  1 frank frank  10 Sep 1 2018 file1
-rw-r--r--  1 frank frank  10 Sep 1 2018 file2
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Music
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Pictures
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Public
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Templates
drwxr-xr-x  2 frank frank 4.0K Apr 1 2018 Videos
```

Para listar todos os arquivos, incluindo arquivos ocultos (aqueles que começam com `.`), use a opção `-a`:

```
$ ls -a
.      .dbus  file1  .profile
..     Desktop file2   Public
.bash_history .dmrc  .gconf .sudo_as_admin_successful
```

Os arquivos de configuração como `.bash_history`, que por padrão ficam ocultos, agora estão visíveis.

Em geral, a sintaxe do comando `ls` é dada por:

```
ls OPTIONS FILE
```

Onde `OPTIONS` é qualquer uma das opções mostradas anteriormente (para ver todas as opções possíveis execute `man ls`), e `FILE` é o nome do arquivo ou detalhes do diretório que você deseja listar.

**NOTE** | Quando `FILE` não está especificado, o diretório atual fica implícito.

## Criar, copiar, mover e remover arquivos

### Criando arquivos com `touch`

O comando `touch` é a maneira mais fácil de criar arquivos novos e vazios. Também pode ser usado para alterar os carimbos de data/hora (ou seja, hora de modificação) dos arquivos e diretórios existentes. A sintaxe para usar `touch` é:

```
touch OPTIONS FILE_NAME(S)
```

Sem nenhuma opção, `touch` cria novos arquivos com os nomes de arquivo fornecidos como argumentos, desde que ainda não existam arquivos com o mesmo nome. `touch` pode criar qualquer número de arquivos simultaneamente:

```
$ touch file1 file2 file3
```

Com esse comando, seriam criados três arquivos novos vazios, chamados `file1`, `file2` e `file3`.

Diversas opções de `touch` foram especificamente pensadas para permitir ao usuário alterar os carimbos de data/hora dos arquivos. Por exemplo, a opção `-a` altera apenas a hora de acesso, enquanto a opção `-m` altera apenas a hora de modificação. O uso de ambas as opções em conjunto altera ambas as horas de acesso e de modificação para o horário atual:

```
$ touch -am file3
```

### Copiando arquivos com `cp`

Como usuários Linux, geralmente copiamos arquivos de um local para outro. Podemos usar o `cp` para todas as tarefas de cópia, seja para mover um arquivo de música ou um arquivo de sistema de um diretório para outro:

```
$ cp file1 dir2
```

Este comando pode ser interpretado literalmente como copiar `file1` para o diretório `dir2`. O resultado é a presença de `file1` dentro de `dir2`. Para que este comando seja executado com sucesso, `file1` deve existir no diretório atual do usuário. Caso contrário, o sistema exibe a mensagem de erro `No such file or directory`.

```
$ cp dir1/file1 dir2
```

Neste caso, observe que o caminho para `file1` é mais explícito. O caminho de origem pode ser expresso como um *caminho relativo* ou um *caminho absoluto*. Os caminhos relativos são dados em referência a um diretório específico, ao passo que os caminhos absolutos não são fornecidos com uma referência. Esclareceremos melhor essa noção mais adiante.

Por enquanto, apenas observe que este comando copia `file1` para o diretório `dir2`. O caminho para `file1` é fornecido com mais detalhes, pois o usuário atualmente não está localizado em `dir1`.

```
$ cp /home/frank/Documents/file2 /home/frank/Documents/Backup
```

Neste terceiro caso, `file2`, localizado em `/home/frank/Documents`, é copiado no diretório `/home/frank/Documents/Backup`. O caminho fornecido aqui é *absoluto*. Nos dois exemplos acima, os caminhos são *relativos*. Quando um caminho começa com o caractere `/`, trata-se de um caminho absoluto; caso contrário, ele é relativo.

A sintaxe geral de `cp` é:

```
cp OPTIONS SOURCE DESTINATION
```

`SOURCE` é o arquivo a ser copiado e `DESTINATION` o diretório para o qual o arquivo será copiado. `SOURCE` e `DESTINATION` podem ser especificados como caminhos absolutos ou relativos.

## Movendo arquivos com o `mv`

Assim como o `cp` para copiar, o Linux fornece um comando para mover e renomear arquivos. Ele se chama `mv`.

A operação de mover é análoga à de recortar e colar que costumamos executar por meio de uma interface gráfica de usuário (GUI).

Se quiser mover um arquivo para um novo local, use `mv` da seguinte maneira:

```
mv FILENAME DESTINATION_DIRECTORY
```

Aqui está um exemplo:

```
$ mv myfile.txt /home/frank/Documents
```

O resultado é que `myfile.txt` é movido para o destino `/home/frank/Documents`.

Para renomear um arquivo, `mv` é usado da seguinte maneira:

```
$ mv old_file_name new_file_name
```

Esse comando muda o nome do arquivo de `old_file_name` para `new_file_name`.

Por padrão, o `mv` não pede confirmação se você quiser sobrescrever (renomear) um arquivo existente. No entanto, podemos fazer com que o sistema exiba uma mensagem, usando a opção `-i`:

```
$ mv -i old_file_name new_file_name  
mv: overwrite 'new_file_name'?
```

Este comando solicita a permissão do usuário antes de sobrescrever `old_file_name` com `new_file_name`.

Inversamente, se usarmos `-f`:

```
$ mv -f old_file_name new_file_name
```

o arquivo seria sobrescrito à força, sem pedir permissão.

## Removendo arquivos com `rm`

`rm` é usado para excluir arquivos. Pense nele como uma forma abreviada da palavra “remover”. Note que a ação de remover um arquivo geralmente é irreversível e, portanto, este comando deve ser usado com cautela.



```
$ rm file1
```

Este comando excluiria `file1`.

```
$ rm -i file1
rm: remove regular file 'file1'?
```

Este comando solicitaria uma confirmação ao usuário antes de excluir `file1`. Lembre-se, vimos a opção `-i` ao usarmos `mv`, acima.

```
$ rm -f file1
```

Este comando exclui `file1` à força, sem pedir sua confirmação.

Podemos excluir vários arquivos ao mesmo tempo:

```
$ rm file1 file2 file3
```

Neste exemplo, `file1`, `file2` e `file3` são excluídos simultaneamente.

A sintaxe de `rm` geralmente é a seguinte:

```
rm OPTIONS FILE
```

## Criando e removendo diretórios

### Criando diretórios com `mkdir`

A criação de diretórios é essencial para organizar seus arquivos e pastas. Os arquivos podem ser agrupados de maneira lógica dentro de um diretório. Para criar um diretório, use `mkdir`:

```
mkdir OPTIONS DIRECTORY_NAME
```

onde `DIRECTORY_NAME` é o nome do diretório a ser criado. Podemos criar qualquer número de diretórios simultaneamente:

```
$ mkdir dir1
```

criaria o diretório `dir1` no diretório atual do usuário.

```
$ mkdir dir1 dir2 dir3
```

O comando anterior criaria três diretórios, `dir1`, `dir2` e `dir3`, ao mesmo tempo.

Para criar um diretório junto com seus subdiretórios, use a opção `-p` (“parents”):

```
$ mkdir -p parents/children
```

Este comando criaria a estrutura de diretórios `parents/children`, ou seja, criaria os diretórios `parents` e `children`. `children` estaria localizado dentro de `parents`.

### Removendo diretórios com o `rmdir`

`rmdir` deleta um diretório *se ele estiver vazio*. Sua sintaxe é dada por:

```
rmdir OPTIONS DIRECTORY
```

onde `DIRECTORY` pode ser um único argumento ou uma lista de argumentos.

```
$ rmdir dir1
```

Este comando excluiria `dir1`.

```
$ rmdir dir1 dir2
```

Este comando excluiria simultaneamente `dir1` e `dir2`.

Podemos remover um diretório junto com seu subdiretório:

```
$ rmdir -p parents/children
```

Isso removeria a estrutura de diretórios `parents/children`. Note que, se algum dos diretórios não estiver vazio, ele não será excluído.

## Manipulação recursiva de arquivos e diretórios

Para manipular um diretório e seu conteúdo, precisamos aplicar a *recursão*. Recursão significa efetuar uma ação e repetir essa ação em toda a árvore de diretórios. No Linux, as opções `-r` ou `-R` ou `--recursive` são geralmente associadas à recursão.

O cenário a seguir ajuda a entender um pouco melhor a recursão:

Você lista o conteúdo de um diretório `students`, que contém dois subdiretórios, `level 1` e `level 2`, e o arquivo chamado `frank`. Aplicando a recursão, o comando `ls` listaria o conteúdo de `alunos`, ou seja, `level 1`, `level 2` e `frank`, mas não terminaria aí. Ele também entraria nos subdiretórios `level 1` e `level 2` para listar seus conteúdos, e assim por diante na árvore de diretórios.

### Listagem recursiva com `ls -R`

`ls -R` é usado para listar o conteúdo de um diretório junto com seus subdiretórios e arquivos.

```
$ ls -R mydirectory
mydirectory/:
file1  newdirectory

mydirectory/newdirectory:
```

Na listagem acima, `mydirectory`, incluindo todo o seu conteúdo, está listado. Podemos observar que `mydirectory` contém o subdiretório `newdirectory` e o arquivo `file1`. `newdirectory` está vazio, por isso nenhum conteúdo é mostrado.

Em geral, para listar o conteúdo de um diretório incluindo seus subdiretórios, usamos:

```
ls -R DIRECTORY_NAME
```

A adição de uma barra a `DIRECTORY_NAME` não tem efeito:

```
$ ls -R animal
```

é similar a

```
$ ls -R animal/
```

### Cópia recursiva com `cp -r`

`cp -r` (ou `-R` ou `--recursive`) permite copiar um diretório junto com todos os seus subdiretórios e arquivos.

```
$ tree mydir
mydir
|_file1
|_newdir
    |_file2
    |_insidenew
    |_lastdir

3 directories, 2 files
$ mkdir newcopy
$ cp mydir newcopy
cp: omitting directory 'mydir'
$ cp -r mydir newcopy
* tree newcopy
newcopy
|_mydir
    |_file1
    |_newdir
        |_file2
        |_insidenew
        |_lastdir

4 directories, 2 files
```

Na listagem acima, observamos que, ao tentar copiar `mydir` em `newcopy`, usando `cp` sem `-r`, o sistema exibe a mensagem `cp: omitting directory 'mydir'`. No entanto, ao adicionar a opção `-r`, todo o conteúdo de `mydir`, incluindo ele mesmo, é copiado para `newcopy`.

Para copiar diretórios e subdiretórios, use:

```
cp -r SOURCE DESTINATION
```

## Remoção recursiva com `rm -r`

`rm -r` remove um diretório e todo o seu conteúdo (subdiretórios e arquivos).

### WARNING

Tenha muito cuidado com o `-r` ou a combinação de opções de `-rf` quando usá-lo com o comando `rm`. Um comando de remoção recursivo em um diretório de sistema importante pode tornar o sistema inutilizável. Empregue o comando de remoção recursiva somente quando tiver certeza absoluta de que o conteúdo de um diretório pode ser removido do computador com segurança.

Ao tentar excluir um diretório sem usar `-r`, o sistema exibe um erro:

```
$ rm newcopy/  
rm: cannot remove 'newcopy/': Is a directory  
$ rm -r newcopy/
```

É necessário adicionar `-r`, como no segundo comando, para que a exclusão tenha efeito.

### NOTE

Você deve estar se perguntando por que não usamos `rmdir` neste caso. Existe uma diferença sutil entre os dois comandos. `rmdir` terá sucesso na exclusão apenas se o diretório fornecido estiver vazio, enquanto `rm -r` pode ser usado independentemente de o diretório estar vazio ou não.

Adicione a opção `-i` para pedir a confirmação antes que o arquivo seja excluído:

```
$ rm -ri mydir/  
rm: remove directory 'mydir/'?
```

O sistema avisa antes de tentar excluir `mydir`.

## Globbing de arquivos e caracteres curinga

O *globbing* de arquivos é um recurso fornecido pelo shell do Unix/Linux para representar múltiplos nomes de arquivo usando caracteres especiais chamados *caracteres curinga*.

Os curingas são, essencialmente, símbolos que podem ser usados para substituir um ou mais caracteres. Eles permitem, por exemplo, mostrar todos os arquivos que começam com a letra `A` ou todos os que terminam com as letras `.conf`.

Os caracteres curinga são utilíssimos, pois podem ser usados com comandos como `cp`, `ls` ou `rm`.

Veja a seguir alguns exemplos de globbing de arquivos:

**`rm *`**

Remove todos os arquivos no diretório de trabalho atual.

**`ls l?st`**

Lista todos os arquivos cujo nome começa com `l`, seguido por qualquer caractere único e terminando com `st`.

**`rmdir [a-z]*`**

Remove todos os diretórios cujo nome começa com uma letra.

## Tipos de caracteres curinga

Existem três caracteres que podem ser usados como curingas no Linux:

### **\* (asterisco)**

representa zero, uma ou mais ocorrências de qualquer caractere.

### **? (interrogação)**

representa uma única ocorrência de qualquer caractere.

### **[ ] (caracteres entre colchetes)**

representa qualquer ocorrência do(s) caractere(s) inseridos nos colchetes. É possível usar diferentes tipos de caracteres, sejam números, letras ou outros caracteres especiais. Por exemplo, a expressão `[0-9]` representa todos os dígitos.

## O asterisco

Um asterisco (\*) corresponde a zero, uma ou mais ocorrências de qualquer caractere.

Por exemplo:

```
$ find /home -name *.png
```

Esse comando localizaria todos os arquivos que terminam com `.png`, como `photo.png`, `cat.png`, `frank.png`. O comando `find` será explorado posteriormente em uma lição seguinte.

Da mesma maneira:

```
$ ls lpic-*.txt
```

listaria todos os arquivos de texto que começam com os caracteres `lpic-` seguidos por qualquer número de caracteres e que terminam com `.txt`, como `lpic-1.txt` e `lpic-2.txt`.

O caractere curinga asterisco pode ser usado para manipular (copiar, excluir ou mover) todo o conteúdo de um diretório:

```
$ cp -r animal/* forest
```

Neste exemplo, todo o conteúdo de `animal` é copiado para `forest`.

Em geral, para copiar todo o conteúdo de um diretório, usamos:

```
cp -r SOURCE_PATH/* DEST_PATH
```

onde `SOURCE_PATH` pode ser omitido se já estivermos no diretório desejado.

O asterisco, assim como qualquer outro caractere curinga, pode ser usado repetidamente no mesmo comando e em qualquer posição:

```
$ rm *ate*
```

Os arquivos com nomes iniciando com zero, uma ou mais ocorrências de qualquer caractere, seguidos das letras `ate` e terminando com zero, uma ou mais ocorrências de qualquer caractere serão removidos.

## O ponto de interrogação

O ponto de interrogação (?) corresponde a uma *única* ocorrência de um caractere.

Considere a listagem:

```
$ ls
last.txt  lest.txt  list.txt  third.txt  past.txt
```

Para retornar apenas os arquivos que começam com `l` seguido por qualquer caractere único e os caracteres `st.txt`, usamos o caractere curinga ponto de interrogação (`?`):

```
$ ls l?st.txt
last.txt    lest.txt    list.txt
```

Apenas os arquivos `last.txt`, `lest.txt` e `list.txt` são exibidos, pois correspondem aos critérios dados.

Da mesma maneira,

```
$ ls ??st.txt
last.txt    lest.txt    list.txt    past.txt
```

exibe os arquivos cujos nomes iniciam com quaisquer dois caracteres seguidos pelo texto `st.txt`.

### Caracteres entre chaves

Os curingas entre colchetes correspondem a qualquer ocorrência do(s) caractere(s) entre colchetes:

```
$ ls l[ae]st.txt
last.txt    lest.txt
```

Este comando listaria todos os arquivos começando com `l` seguido por *qualquer um* dos caracteres do conjunto `ae` e terminando com `st.txt`.

Os colchetes também podem indicar intervalos:

```
$ ls l[a-z]st.txt
last.txt    lest.txt    list.txt
```

Esse comando exibe todos os arquivos com nomes começando com `l` seguido por qualquer letra minúscula no intervalo de `a` a `z` e terminando com `st.txt`.

Também podemos definir múltiplos intervalos entre colchetes:

```
$ ls
student-1A.txt student-2A.txt student-3.txt
```



```
$ ls student-[0-9][A-Z].txt
student-1A.txt student-2A.txt
```

A lista mostra um diretório escolar com uma lista de alunos registrados. Para listar apenas os alunos cujos números de registro, é preciso atender aos seguintes critérios:

- começa com `student-`
- seguido por um número e um caractere em maiúscula
- e termina com `.txt`

### Combinando caracteres curinga

Os caracteres curinga podem ser combinados, como em:

```
$ ls
last.txt  lest.txt  list.txt  third.txt  past.txt
$ ls [plf]?st*
last.txt  lest.txt  list.txt  past.txt
```

O primeiro componente curinga (`[plf]`) corresponde a qualquer um dos caracteres `p`, `l` ou `f`. O segundo componente curinga (`?`) corresponde a qualquer caractere único. O terceiro componente curinga (`*`) corresponde a zero, uma ou múltiplas ocorrências de qualquer caractere.

```
$ ls
file1.txt file.txt file23.txt fom23.txt
$ ls f*[0-9].txt
file1.txt file23.txt fom23.txt
```

O comando anterior exibe todos os arquivos que começam com a letra `f`, seguido por qualquer conjunto de letras, pelo menos uma ocorrência de um dígito e termina com `.txt`. Observe que `file.txt` não é exibido, pois não corresponde a esses critérios.

## Exercícios Guiados

1. Considere a listagem abaixo:

```
$ ls -lh
total 60K
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Desktop
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Documents
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Downloads
-rw-r--r-- 1 frank frank 21 Sep 7 12:59 emp_name
-rw-r--r-- 1 frank frank 20 Sep 7 13:03 emp_salary
-rw-r--r-- 1 frank frank 8.8K Apr 1 2018 examples.desktop
-rw-r--r-- 1 frank frank 10 Sep 1 2018 file1
-rw-r--r-- 1 frank frank 10 Sep 1 2018 file2
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Music
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Pictures
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Public
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Templates
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Videos
```

- O que o caractere `d` representa na saída?

- Por que os tamanhos são mostrados no formato legível por humanos?

- Qual seria a diferença na saída se `ls` fosse usado sem argumento?

2. Considere o comando abaixo:

```
$ cp /home/frank/emp_name /home/frank/backup
```

- O que aconteceria ao arquivo `emp_name` se esse comando fosse executado com sucesso?

- Se `emp_name` fosse um diretório, qual opção precisaria ser adicionada a `cp` para executar o comando?

- Se `cp` fosse alterado para `mv`, quais seriam os resultados esperados?

3. Considere a listagem:

```
$ ls  
file1.txt file2.txt file3.txt file4.txt
```

Qual caractere curinga ajudaria a remover todo o conteúdo deste diretório?

4. Com base na listagem anterior, quais arquivos seriam exibidos com o comando a seguir?

```
$ ls file*.txt
```

5. Complete o comando adicionando os dígitos e caracteres entre os colchetes, de modo listar todo o conteúdo acima:

```
$ ls file[.].txt
```

## Exercícios Exploratórios

1. Em seu diretório inicial, crie arquivos chamados `dog` e `cat`.
2. Ainda no diretório inicial, crie um diretório chamado `animal`. Mova `dog` e `cat` para dentro de `animal`.
3. Vá à pasta `Documents` em seu diretório inicial e, dentro dela, crie o diretório `backup`.
4. Copie `animal` e seu conteúdo para `backup`.
5. Renomeie `animal` em `backup` como `animal.bkup`.
6. O diretório `/home/lpi/databases` contém muitos arquivos, dentre os quais: `db-1.tar.gz`, `db-2.tar.gz` e `db-3.tar.gz`. Qual comando podemos usar para listar apenas os arquivos mencionados acima?

7. Considere a listagem:

```
$ ls  
cne1222223.pdf cne12349.txt cne1234.pdf
```

Usando um único caractere de globbing, qual comando removeria apenas os arquivos `pdf`?

## Resumo

Nesta lição, exploramos como visualizar o que está dentro de um diretório com o comando `ls`, como copiar (`cp`) arquivos e pastas e como movê-los (`mv`). Também vimos como novos diretórios podem ser criados com o comando `mkdir`. Os comandos para remover arquivos (`rm`) e pastas (`rmdir`) também foram abordados.

Nesta lição, você também aprendeu sobre o globbing de arquivos e caracteres curinga. O globbing de arquivo é usado para representar vários nomes de arquivo usando caracteres especiais chamados curingas. Estes são os caracteres curinga básicos e seus significados:

### **? (interrogação)**

representa uma única ocorrência de qualquer caractere.

### **[ ] (caracteres entre colchetes)**

representa qualquer ocorrência do(s) caractere(s) inseridos nos colchetes.

### **\* (asterisco)**

representa zero, uma ou mais ocorrências de qualquer caractere.

Podemos incluir qualquer combinação desses caracteres curinga na mesma instrução.

## Respostas aos Exercícios Guiados

1. Considere a listagem abaixo:

```
$ ls -lh
total 60K
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Desktop
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Documents
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Downloads
-rw-r--r-- 1 frank frank 21 Sep 7 12:59 emp_name
-rw-r--r-- 1 frank frank 20 Sep 7 13:03 emp_salary
-rw-r--r-- 1 frank frank 8.8K Apr 1 2018 examples.desktop
-rw-r--r-- 1 frank frank 10 Sep 1 2018 file1
-rw-r--r-- 1 frank frank 10 Sep 1 2018 file2
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Music
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Pictures
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Public
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Templates
drwxr-xr-x 2 frank frank 4.0K Apr 1 2018 Videos
```

- O que o caractere `d` representa na saída?

`d` é o caractere que identifica um diretório.

- Por que os tamanhos são mostrados no formato legível por humanos?

Por causa da opção `-h`.

- Qual seria a diferença na saída se `ls` fosse usado sem argumento?

Seriam mostrados apenas os nomes dos diretórios e arquivos.

2. Considere o comando abaixo:

```
$ cp /home/frank/emp_name /home/frank/backup
```

- O que aconteceria ao arquivo `emp_name` se esse comando fosse executado com sucesso?

`emp_name` seria copiado em `backup`.

- Se `emp_name` fosse um diretório, qual opção precisaria ser adicionada a `cp` para executar o

comando?

`-r`

- Se `cp` fosse alterado para `mv`, quais seriam os resultados esperados?

`emp_name` seria movido para `backup`. Ele não estaria mais presente no diretório inicial do usuário `frank`.

3. Considere a listagem:

```
$ ls
file1.txt file2.txt file3.txt file4.txt
```

Qual caractere curinga ajudaria a remover todo o conteúdo deste diretório?

O asterisco `*`.

4. Com base na listagem anterior, quais arquivos seriam exibidos com o comando a seguir?

```
$ ls file*.txt
```

Todos eles, já que o asterisco representa qualquer número de caracteres.

5. Complete o comando adicionando os dígitos e caracteres entre os colchetes, de modo listar todo o conteúdo acima:

```
$ ls file[0-9].txt
```

`file[0-9].txt`

## Respostas aos Exercícios Exploratórios

1. Em seu diretório inicial, crie arquivos chamados `dog` e `cat`.

```
$ touch dog cat
```

2. Ainda no diretório inicial, crie um diretório chamado `animal`. Mova `dog` e `cat` para dentro de `animal`.

```
$ mkdir animal  
$ mv dog cat -t animal/
```

3. Vá à pasta `Documents` em seu diretório inicial e, dentro dela, crie o diretório `backup`.

```
$ cd ~/Documents  
$ mkdir backup
```

4. Copie `animal` e seu conteúdo para `backup`.

```
$ cp -r animal ~/Documents/backup
```

5. Renomeie `animal` em `backup` como `animal.bkup`.

```
$ mv animal/ animal.bkup
```

6. O diretório `/home/lpi/databases` contém muitos arquivos, dentre os quais: `db-1.tar.gz`, `db-2.tar.gz` e `db-3.tar.gz`. Qual comando podemos usar para listar apenas os arquivos mencionados acima?

```
$ ls db-[1-3].tar.gz
```

7. Considere a listagem:

```
$ ls  
cne1222223.pdf cne12349.txt cne1234.pdf
```



Usando um único caractere de globbing, qual comando removeria apenas os arquivos pdf?

```
$ rm *.pdf
```



## 103.3 Lição 2

<b>Certificação:</b>	LPIC-1
<b>Versão:</b>	5.0
<b>Tópico:</b>	103 Comandos GNU e Unix
<b>Objetivo:</b>	103.3 Gerenciamento básico de arquivos
<b>Lição:</b>	2 de 2

### Introdução

#### Como encontrar arquivos

Conforme você usa sua máquina, os arquivos vão aumentando progressivamente em número e tamanho. Às vezes, é difícil localizar um arquivo específico. Felizmente, o Linux inclui o `find` para pesquisar e localizar arquivos rapidamente. O `find` usa a seguinte sintaxe:

```
find STARTING_PATH OPTIONS EXPRESSION
```

#### STARTING\_PATH

define o diretório em que a pesquisa se inicia.

#### OPTIONS

controla o comportamento e adiciona critérios específicos para otimizar o processo de busca.

#### EXPRESSION

define os termos da pesquisa.

```
$ find . -name "myfile.txt"
./myfile.txt
```

O caminho inicial, neste caso, é o diretório atual. A opção `-name` especifica que a pesquisa é baseada no nome do arquivo. `myfile.txt` é o nome do arquivo a ser pesquisado. Ao usar globbing de arquivo, inclua sempre a expressão entre aspas:

```
$ find /home/frank -name "*.png"
/home/frank/Pictures/logo.png
/home/frank/screenshot.png
```

Este comando busca por todos os arquivos que terminam com `.png`, começando no diretório `/home/frank/` e abaixo dele. Se você não entendeu o uso do asterisco (`*`), ele foi abordado na lição anterior.

### Usando critérios para acelerar a pesquisa

Use `find` para localizar arquivos com base em *tipo*, *tamanho* ou *hora*. Se especificarmos uma ou mais opções, os resultados desejados são obtidos em menos tempo.

As opções para localizar arquivos com base no tipo incluem:

#### `-type f`

busca por arquivos.

#### `-type d`

busca por diretórios.

#### `-type l`

busca por links simbólicos.

```
$ find . -type d -name "example"
```

Este comando procura por todos os diretórios, no diretório atual e abaixo dele, que tenham o nome `example`.

Dentre os outros critérios que podem ser usados com `find`, temos:

**-name**

pesquisa com base no nome fornecido.

**-iname**

pesquisa com base no nome, desconsiderando maiúsculas e minúsculas (ou seja, o caso de teste `myFile` é semelhante a `MYFILE`).

**-not**

retorna os resultados que *não* correspondem ao caso de teste.

**-maxdepth N**

pesquisa no diretório atual, além dos subdiretórios até N níveis de profundidade.

**Localizando arquivos por hora de modificação**

`find` também permite filtrar uma hierarquia de diretórios com base em quando o arquivo foi modificado:

```
$ sudo find / -name "*.conf" -mtime 7
/etc/logrotate.conf
```

Este comando procura por todos os arquivos em todo o sistema de arquivos (o caminho inicial é o diretório raiz, ou seja, `/`) que terminam com os caracteres `.conf` e que foram modificados nos últimos sete dias. Este comando exige privilégios elevados para acessar diretórios na base da estrutura de diretórios do sistema, daí o uso de `sudo` neste caso. O argumento passado para `mtime` representa o *número de dias* desde a última modificação do arquivo.

**Localizando arquivos por tamanho**

O `find` também pode localizar arquivos por *tamanho*. Por exemplo, se quisermos encontrar arquivos maiores que 2G em `/var`:

```
$ sudo find /var -size +2G
/var/lib/libvirt/images/debian10.qcow2
/var/lib/libvirt/images/rhel8.qcow2
```

A opção `-size` exibe arquivos de tamanhos correspondentes ao argumento passado. Eis alguns exemplos de argumentos:

**-size 100b**

arquivos com exatamente 100 bytes.

**-size +100k**

arquivos maiores que 100 kilobytes.

**-size -20M**

arquivos menores que 20 megabytes.

**-size +2G**

arquivos maiores que 2 gigabytes.

**NOTE**

Para encontrar arquivos vazios, podemos usar: `find . -size 0b` ou `find . -empty`.

**O que fazer com os resultados**

Uma vez que a pesquisa é feita, é possível realizar uma ação no conjunto de resultados usando `-exec`:

```
$ find . -name "*.conf" -exec chmod 644 '{}' \;
```

Esse comando filtra todos os objetos no diretório atual (.) e abaixo dele para nomes de arquivo terminando com `.conf` e em seguida executa o comando `chmod 644` para modificar as permissões de arquivo nos resultados.

Por enquanto, não se preocupe com o significado de `'{}' \;`, pois isso será discutido mais adiante.

**Usando o `grep` para filtrar por arquivos com base no conteúdo**

`grep` é usado para buscar pela ocorrência de uma palavra-chave.

Considere uma situação na qual precisamos encontrar arquivos com base no conteúdo:

```
$ find . -type f -exec grep "lpi" '{}' \; -print
./bash_history
Alpine/M
helping/M
```

Esse comando busca, na hierarquia de diretórios atual (.), por objetos que são arquivos (`-type f`) e

em seguida executa o comando `grep "lpi"` para cada arquivo que satisfaça as condições. Os arquivos que atendem a essas condições são impressos na tela (`-print`). As chaves (`{}`) servem para reservar o espaço para os resultados encontrados por `find`. As `{}` são postas entre aspas simples (`'`) para evitar passar arquivos com nomes contendo caracteres especiais para o `grep`. O comando `-exec` é concluído com um ponto e vírgula (`;`), que deve ser escapado (`\;`) para não ser interpretado pelo shell.

A opção `-delete`, se colocada no final de uma expressão, excluiria todos os arquivos correspondentes à descrição. Esta opção deve ser usada quando você tiver certeza de que os resultados correspondem apenas aos arquivos que deseja excluir.

No exemplo abaixo, `find` localiza todos os arquivos na hierarquia começando no diretório atual e, em seguida, exclui todos os arquivos que terminam com os caracteres `.bak`:

```
$ find . -name "*.bak" -delete
```

## Arquivos de pacote

### O comando `tar` (Arquivamento e compactação)

O comando `tar`, abreviação de “tape archive(r)”, é usado para criar arquivos tar convertendo um grupo de arquivos em um pacote. Os arquivos de pacote são úteis para mover ou fazer backup de um grupo de arquivos facilmente. Pense no `tar` como uma ferramenta que cria uma cola na qual os arquivos podem ser grudados, agrupados e facilmente movidos.

O `tar` também tem a capacidade de extrair arquivos tar, exibir uma lista dos arquivos incluídos no pacote e adicionar mais arquivos a um pacote existente.

A sintaxe do comando `tar` é a seguinte:

```
tar [OPERATION_AND_OPTIONS] [ARCHIVE_NAME] [FILE_NAME(S)]
```

#### OPERATION

Somente um argumento de operação é permitido e exigido. As operações mais frequentemente usadas são:

##### **`--create (-c)`**

Cria um novo arquivo tar.

**--extract (-x)**

Extrai o pacote inteiro ou um ou mais arquivos de um pacote.

**--list (-t)**

Exibe uma lista dos arquivos incluídos no pacote.

**OPTIONS**

As opções usadas com mais frequência são:

**--verbose (-v)**

Mostra os arquivos que estão sendo processados pelo comando `tar`.

**--file=archive=name (-f archive-name)**

Especifica o nome de arquivo do pacote.

**ARCHIVE\_NAME**

O nome do arquivo de pacote.

**FILE\_NAME(S)**

Uma lista separada por espaços com os nomes de arquivos a serem extraídos. Se não estiver presente, o pacote inteiro é extraído.

**Criando um arquivo de pacote**

Digamos que temos um diretório chamado `stuff` no diretório atual e queremos salvá-lo em um arquivo chamado `archive.tar`. Executaríamos para isso o seguinte comando:

```
$ tar -cvf archive.tar stuff
stuff/
stuff/service.conf
```

Eis o que essas opções significam de fato:

**-c**

Cria um arquivo de pacote.

**-v**

Exibe o progresso no terminal enquanto o arquivo de pacote é criado. Também chamado de modo “verboso”. O `-v` sempre é opcional nesses comandos, mas é útil.

**-f**

Permite especificar o nome de arquivo do pacote.

Em geral, para arquivar um único diretório ou um único arquivo no Linux, usamos:

```
tar -cvf NAME-OF-ARCHIVE.tar /PATH/TO/DIRECTORY-OR-FILE
```

**NOTE**

O tar funciona de maneira recursiva. Ele realiza a ação solicitada em todos os diretórios subsequentes dentro do diretório especificado.

Para empacotar diversos diretórios de uma vez só, listamos todos eles delimitando-os por um espaço na seção `/PATH/TO/DIRECTORY-OR-FILE`:

```
$ tar -cvf archive.tar stuff1 stuff2
```

Isso cria um arquivo de pacote com `stuff1` e `stuff2` em `archive.tar`

**Extraindo um pacote**

Podemos extrair um arquivo de pacote usando o `tar`:

```
$ tar -xvf archive.tar
stuff/
stuff/service.conf
```

Isso extrai o conteúdo de `archive.tar` para o diretório atual.

Este comando é igual ao comando de criação de pacotes usado acima, exceto porque a opção `-x` substitui a opção `-c`.

Para extrair o conteúdo do pacote para um diretório específico, usamos `-C`:

```
$ tar -xvf archive.tar -C /tmp
```

Isso extrai o conteúdo de `archive.tar` para o diretório `/tmp`.

```
$ ls /tmp
```



```
stuff
```

## Compactando com o tar

O comando `tar` do GNU incluído nas distribuições Linux pode criar um arquivo `.tar` e, em seguida, compactá-lo com a compactação `gzip` ou `bzip2` em um único comando:

```
$ tar -czvf name-of-archive.tar.gz stuff
```

Este comando criaria um arquivo compactado usando o algoritmo `gzip` (`-z`).

Embora a compressão `gzip` seja mais freqüentemente usada para criar arquivos `.tar.gz` ou `.tgz`, o `tar` também suporta a compressão `bzip2`. Isso permite a criação de arquivos compactados `bzip2`, geralmente chamados de arquivos `.tar.bz2`, `.tar.bz` ou `.tbz`.

Para isso, substituímos `-z`, de `gzip`, por `-j`, de `bzip2`:

```
$ tar -cjvf name-of-archive.tar.bz stuff
```

Para descompactar o arquivo, substituímos `-c` por `-x`, onde `x` significa “extract”:

```
$ tar -xzf archive.tar.gz
```

O `gzip` é mais rápido, mas geralmente compacta um pouco menos, de modo que o arquivo obtido é um pouco maior. O `bzip2` é mais lento, mas comprime um pouco mais, então o arquivo fica um pouco menor. Em geral, porém, `gzip` e `bzip2` são praticamente a mesma coisa; ambos funcionam de forma semelhante.

Outra alternativa seria aplicar a compressão `gzip` ou `bzip2` usando o comando `gzip` para as compressões `gzip` e `bzip` para as compressões `bzip`. Por exemplo, para aplicar a compactação `gzip`, use:

```
gzip FILE-TO-COMPRESS
```

## gzip

cria o arquivo compactado com o mesmo nome, mas com a terminação `.gz`.

## gzip

remove os arquivos originais após criar o arquivo compactado.

O comando `bzip2` funciona de maneira semelhante.

Para descompactar os arquivos usamos `gunzip` ou `bunzip2`, dependendo do algoritmo usado para compactá-los.

## O comando cpio

`cpio` significa “copy in, copy out”. É usado para processar arquivo de pacote como os arquivos `*.cpio` ou `*.tar`.

O `cpio` executa as seguintes operações:

- Copiar arquivos para um pacote.
- Extrair arquivos de um pacote.

Ele usa a lista de arquivos da entrada padrão (principalmente a saída de `ls`).

Para criar um arquivo `cpio`, usamos:

```
$ ls | cpio -o > archive.cpio
```

A opção `-o` instrui o `cpio` a criar uma saída. Neste caso, o arquivo de saída criado é `archive.cpio`. O comando `ls` lista o conteúdo do diretório atual que será empacotado.

Para extrair o arquivo de pacote, usamos:

```
$ cpio -id < archive.cpio
```

A opção `-i` é usada para realizar a extração. A opção `-d` cria a pasta de destino. O caractere `<` representa a entrada padrão. O arquivo de entrada a ser extraído é `archive.cpio`.

## O comando dd

O `dd` copia dados de um local para outro. A sintaxe de linha de comando de `dd` difere de muitos outros programas Unix, pois ele usa a sintaxe `option = value` para as opções de linha de comando ao invés dos formatos padrão GNU `-option value` ou `--option=value`:

```
$ dd if=oldfile of=newfile
```

Este comando copia o conteúdo de `oldfile` para `newfile`, onde `if=` é o arquivo de entrada e `of=` refere-se ao arquivo de saída.

**NOTE**

O comando `dd` normalmente não exibe nada na tela até que o comando seja concluído. Ao fornecer a opção `status=progress`, o console exibe o andamento do trabalho realizado pelo comando. Por exemplo: `dd status=progress if=oldfile of=newfile`.

O `dd` também é usado para alterar dados para maiúsculas/minúsculas ou para escrever diretamente em dispositivos de bloco como `/dev/sdb`:

```
$ dd if=oldfile of=newfile conv=ucase
```

Esse comando copiaria todo o conteúdo de `oldfile` para `newfile` e colocaria todo o texto em maiúsculas.

O comando a seguir faria backup do disco rígido inteiro localizado em `/dev/sda` para um arquivo de nome `backup.dd`:

```
$ dd if=/dev/sda of=backup.dd bs=4096
```

## Exercícios Guiados

1. Considere a listagem a seguir:

```
$ find /home/frank/Documents/ -type d
/home/frank/Documents/
/home/frank/Documents/animal
/home/frank/Documents/animal/domestic
/home/frank/Documents/animal/wild
```

- Que tipo de arquivos esse comando produziria?

- A busca começaria em qual diretório?

2. Um usuário deseja compactar sua pasta de backup. Ele usa o seguinte comando:

```
$ tar cvf /home/frank/backup.tar.gz /home/frank/dir1
```

Qual opção está faltando para compactar o backup usando o algoritmo `gzip`?

## Exercícios Exploratórios

1. O administrador do sistema precisa realizar verificações regulares para remover arquivos volumosos. Esses arquivos volumosos estão localizados em `/var` e terminam com uma extensão `.backup`.

- Escreva o comando, usando `find`, para localizar esses arquivos:

- Uma análise do tamanhos desses arquivos revela que eles variam de `100M` a `1000M`. Complete o comando anterior com esta nova informação para poder localizar os arquivos de backup variando de `100M` a `1000M`:

- Finalmente, complete este comando com a ação delete para remover esses arquivos:

2. No diretório `/var`, existem quatro arquivos de backup:

```
db-jan-2018.backup  
db-feb-2018.backup  
db-march-2018.backup  
db-apr-2018.backup
```

- Usando o `tar`, especifique o comando usado para criar um arquivo de pacote com o nome `db-first-quarter-2018.backup.tar`:

- Usando o `tar`, especifique o comando usado para criar o arquivo de pacote e compactá-lo usando o `gzip`. Note que o nome do arquivo resultante deve terminar com `.gz`:

## Resumo

Nesta seção, você aprendeu:

- Como encontrar arquivos com `find`.
- Como adicionar critérios de busca com base em hora, tipo de arquivo e tamanho fornecendo argumentos ao `find`.
- O que fazer com os resultados.
- Como arquivar, compactar e descompactar arquivos usando `tar`.
- Processamento de arquivos de pacote com `cpio`.
- Cópia de arquivos com `dd`.

## Respostas aos Exercícios Guiados

1. Considere a listagem a seguir:

```
$ find /home/frank/Documents/ -type d
/home/frank/Documents/
/home/frank/Documents/animal
/home/frank/Documents/animal/domestic
/home/frank/Documents/animal/wild
```

- Que tipo de arquivos esse comando produziria?

Diretórios.

- A busca começaria em qual diretório?

/home/frank/Documents

2. Um usuário deseja compactar sua pasta de backup. Ele usa o seguinte comando:

```
$ tar cvf /home/frank/backup.tar.gz /home/frank/dir1
```

Qual opção está faltando para compactar o backup usando o algoritmo `gzip`?

A opção `-z`.

## Respostas aos Exercícios Exploratórios

1. O administrador do sistema precisa realizar verificações regulares para remover arquivos volumosos. Esses arquivos volumosos estão localizados em `/var` e terminam com uma extensão `.backup`.

- Escreva o comando, usando `find`, para localizar esses arquivos:

```
$ find /var -name *.backup
```

- Uma análise do tamanho desses arquivos revela que eles variam de `100M` a `1G`. Complete o comando anterior com esta nova informação para poder localizar os arquivos de backup variando de `100M` a `1G`:

```
$ find /var -name *.backup -size +100M -size -1000M
```

- Finalmente, complete este comando com a ação `delete` para remover esses arquivos:

```
$ find /var -name *.backup -size +100M -size -1000M -delete
```

2. No diretório `/var`, existem quatro arquivos de backup:

```
db-jan-2018.backup  
db-feb-2018.backup  
db-march-2018.backup  
db-apr-2018.backup
```

- Usando o `tar`, especifique o comando usado para criar um arquivo de pacote com o nome `db-first-quarter-2018.backup.tar`:

```
$ tar -cvf db-first-quarter-2018.backup.tar db-jan-2018.backup db-feb-  
2018.backup db-march-2018.backup db-apr-2018.backup
```

- Usando o `tar`, especifique o comando usado para criar o arquivo de pacote e compactá-lo usando o `gzip`. Note que o nome do arquivo resultante deve terminar com `.gz`:

```
$ tar -zcvf db-first-quarter-2018.backup.tar.gz db-jan-2018.backup db-feb-
```



```
2018.backup db-march-2018.backup db-apr-2018.backup
```