TRABALHANDO COM ARQUIVOS E DIRETÓRIOS

6.1 Introdução

Ao trabalhar em um sistema operacional Linux, você precisará saber como manipular arquivos e diretórios. Algumas distribuições do Linux têm aplicativos baseados em GUI que permitem gerenciar arquivos, mas é importante saber como executar essas operações por meio da linha de comando.

A linha de comando apresenta uma coleção rica de comandos que permitem gerenciar arquivos. Neste capítulo, você aprenderá como listar arquivos em um diretório, além de copiar, mover e excluir arquivos.

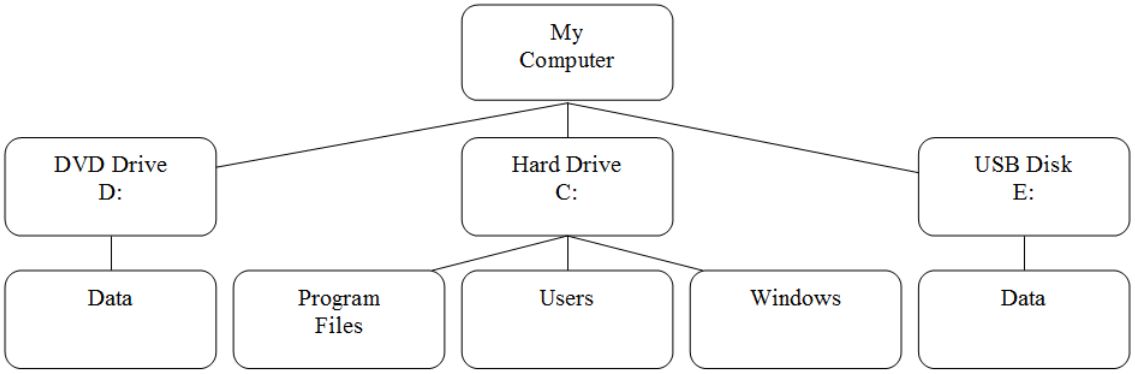
Os principais conceitos ensinados neste capítulo serão expandidos em capítulos posteriores à medida que mais comandos de manipulação de arquivos forem abordados, como a visualização de arquivos, compactação de arquivos e definição de permissões de arquivos.

6.2 Entendendo Arquivos e Diretórios

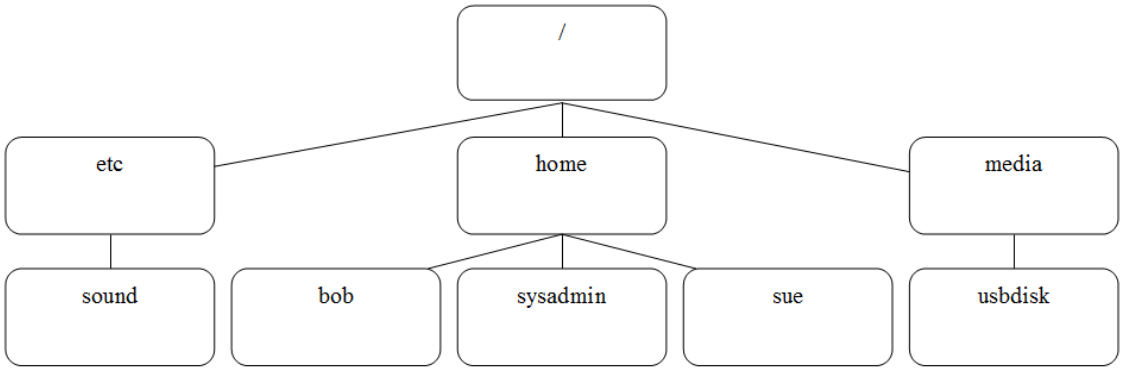
Os arquivos são usados para armazenar dados, como texto, gráficos e programas. Diretórios (também conhecido como, "pastas") são usados para fornecer uma estrutura organizacional hierárquica. Essa estrutura é um pouco diferente do que você pode estar acostumado se já trabalhou anteriormente em sistemas Microsoft Windows.

Em um sistema Windows, o nível superior da estrutura de diretórios é chamado *Meu Computador*. Cada dispositivo físico (disco rígido, unidade de DVD, pendrive USB, unidade de rede, etc.) aparece em *Meu computador*, cada um com uma letra de unidade, como *C:* ou *D:*. Uma representação visual dessa estrutura:

As estruturas de diretório mostradas abaixo são fornecidas apenas como exemplos. Esses diretórios podem não estar presentes no ambiente de máquina virtual deste curso.



Como o Windows, uma estrutura de diretórios do Linux tem um nível superior. No entanto, não é chamado *Meu Computador*, mas sim o diretório raiz e é simbolizado pelo caractere `/`. Também não há drives no Linux; cada dispositivo físico é acessível em um diretório, não em uma letra de unidade. Uma representação visual de uma estrutura de diretórios típica do Linux:



Essa estrutura de diretórios é chamada de *sistema de arquivos* (*filesystem*) pela maioria dos usuários do Linux.

Para visualizar o filesystem raiz, digite `ls /`:

```
sysadmin@localhost:~$ ls /
bin   dev  home  lib    media  opt   root  sbin     selinux  sys  usr
boot  etc  init  lib64  mnt    proc  run   sbin???  srv      tmp  var
```

Observe que há muitos diretórios descritivos, incluindo `/boot`, que contém arquivos para inicializar o computador.

6.2.1 Caminho de Diretório

Usando o gráfico na seção anterior como um ponto de referência, você verá que existe um diretório chamado `sound` sob um diretório chamado `etc`, que está sob o diretório `/`. Uma maneira mais fácil de dizer isso é se referir ao *path* (caminho).

Using the graphic in the previous section as a point of reference, you will see that there is a directory named `sound` under a directory named `etc`, which is under the `/` directory. An easier way to say this, is to refer to the *path*.

Considere isto:

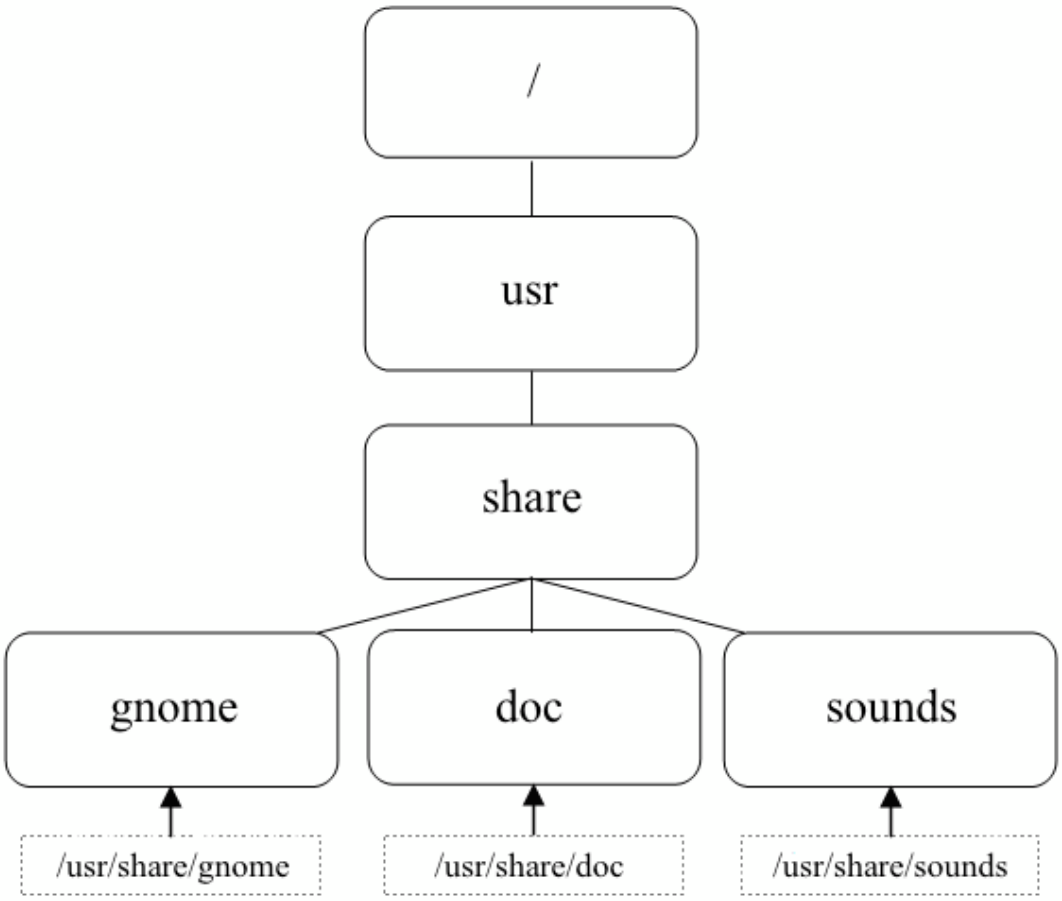
O diretório **`/etc`** originalmente significava “et cetera” no início da documentação do Bell Labs e costumava conter arquivos que não pertenciam a outro lugar. Nas distribuições modernas do Linux, o diretório **`/etc`** normalmente contém arquivos de configuração estáticos, conforme definido pelo File Hierarchy Standard (FHS).

Um caminho permite que você especifique a localização exata de um diretório. Para o diretório de `sound`, o caminho seria `/etc/sound`. O primeiro caractere `/` representa o diretório `raiz`, enquanto cada outro `/` caractere é usado para separar os nomes dos diretórios.

Esse tipo de caminho é chamado de *caminho absoluto*. Com um caminho absoluto, você sempre fornece instruções para um diretório (ou um arquivo) a partir da parte superior da estrutura de diretórios, o diretório `raiz`. Mais adiante neste capítulo, abordaremos um tipo diferente de caminho chamado caminho relativo.

Nota: As estruturas de diretório mostradas abaixo são fornecidas apenas como exemplos. Esses diretórios podem não estar presentes no ambiente de máquina virtual deste curso.

O gráfico a seguir demonstra três caminhos absolutos adicionais:


/
usr
share
gnome
doc
sounds
/usr/share/gnome
/usr/share/doc
/usr/share/sounds

6.2.2 Diretório Home

O termo *diretório home* geralmente causa confusão ao iniciar usuários do Linux. Para começar, na maioria das distribuições Linux existe um diretório chamado `home` no diretório raiz: `/home`.

Sob este diretório `/home`, haverá um diretório para cada usuário no sistema. O nome do diretório será o mesmo que o nome do usuário, portanto, um usuário chamado "bob" teria um *diretório home* chamado `/home/bob`.

Seu diretório home é um diretório muito importante. Para começar, quando você abre um shell, você deve ser colocado automaticamente em seu diretório home, pois é onde você fará a maior parte do seu trabalho.

Além disso, seu diretório home é um dos poucos diretórios onde você tem o controle total para criar e excluir arquivos e diretórios adicionais. A maioria dos outros diretórios em um sistema de arquivos do Linux é protegida com permissões de arquivo, um tópico que será abordado em detalhes em um capítulo posterior.

Na maioria das distribuições Linux, os únicos usuários que podem acessar qualquer arquivo em seu diretório home são você e o administrador no sistema (o usuário root). Isso pode ser alterado usando permissões de arquivo.

Seu diretório pessoal tem até mesmo um símbolo especial que você pode usar para representá-lo: `~`. Se o seu diretório home for `/home/sysadmin`, você pode simplesmente digitar `~` na linha de comando no lugar de `/home/sysadmin`. Você também pode consultar o diretório home de outro usuário usando a notação `~`*`user`*, em que *`user`* é o nome da conta do usuário cujo diretório homel você deseja consultar. Por exemplo, `~bob` seria o mesmo que `/home/bob`. Aqui, vamos mudar para o diretório home do usuário:

```
sysadmin@localhost:~$ cd ~
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$
```

Observe que uma listagem revela os subdiretórios contidos no diretório home. Alterar diretórios requer atenção aos detalhes:

```
sysadmin@localhost:~$ cd downloads
-bash: cd: downloads: No such file or directory
sysadmin@localhost:~$
```

Por que o comando acima resultou em um erro? Isso ocorre porque os ambientes Linux são sensíveis a letras maiúsculas e minúsculas. Mudar para o diretório `Downloads` requer a grafia correta - incluindo a letra maiúscula `D`:

```
sysadmin@localhost:~$ cd Downloads
sysadmin@localhost:~/Downloads$
```

6.2.3 Diretório atual

Seu *diretório atual* é o diretório em que você está trabalhando atualmente em um terminal. Quando você abre um terminal pela primeira vez, o diretório atual deve ser seu diretório home, mas isso pode mudar conforme você explora o sistema de arquivos e muda para outros diretórios.

Enquanto você estiver em um ambiente de linha de comando, poderá determinar seu diretório atual usando o comando `pwd`:

```
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

Além disso, a maioria dos sistemas tem o prompt de usuário padrão para exibir o diretório atual:

```
[sysadmin@localhost ~]$
```

No gráfico acima, o caractere `~` indica seu diretório atual. Como mencionado anteriormente, o caractere `~` representa seu diretório home.

Normalmente, o prompt exibe apenas o nome do diretório atual, não o caminho completo do diretório raiz para baixo. Em outras palavras, se você estivesse no diretório `/usr/share/doc`, seu prompt provavelmente forneceria apenas o nome `doc` para o diretório atual. Se você quiser o caminho completo, use o comando `pwd`.

6.2.4 Mudando Diretórios

Se você quiser mudar para um diretório diferente, use o comando `cd` (change directory). Por exemplo, o seguinte comando irá alterar o diretório atual para um diretório chamado `/etc/sound/events`:

```
sysadmin@localhost:~$ cd /etc/sound/events
sysadmin@localhost:/etc/sound/events$
```

Note que não há saída se o comando `cd` for bem sucedido. Este é um daqueles tipos de coisas que onde "não há notícias, são boas". Se você tentar mudar para um diretório que não existe, você receberá uma mensagem de erro:

```
sysadmin@localhost:/etc/sound/events$ cd /etc/junk
-bash: cd: /etc/junk: No such file or directory
sysadmin@localhost:/etc/sound/events$
```

Se você quiser retornar ao seu diretório home, você pode digitar o comando `cd` sem argumentos ou usar o comando `cd` com o caractere `~` como um argumento:

```
sysadmin@localhost:/etc/sound/events$ cd
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$ cd /etc
sysadmin@localhost:/etc$ cd ~
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

6.2.5 Caminhos absolutos vs. relativos

Lembre-se de que um nome de caminho é essencialmente uma descrição de onde um arquivo ou diretório está localizado no sistema de arquivos. Você também pode considerar um nome de caminho como direções que informam ao sistema onde encontrar um arquivo ou diretório. Por exemplo, o comando `cd /etc/perl/Net` significa "mude para o diretório `Net`, que você encontrará no diretório `perl`, que você encontrará no diretório `etc`, que você encontrará no diretório `/`.

Quando você dá um nome de caminho que começa no diretório raiz, ele é chamado de *caminho absoluto*. Em muitos casos, fornecer um caminho absoluto faz sentido. Por exemplo, se você está em seu diretório `home` e deseja ir para o diretório `/etc/perl/Net`, fornecer um caminho absoluto para o comando `cd` faz sentido:

```
sysadmin@localhost:~$ cd /etc/perl/Net
sysadmin@localhost:/etc/perl/Net$
```

No entanto, e se você estivesse no diretório `/etc/perl` e quisesse ir para o diretório `/etc/perl/Net`? Seria tedioso digitar o caminho completo para chegar a um diretório que é apenas um nível abaixo da sua localização atual. Em uma situação como essa, você deseja usar um *caminho relativo*:

```
sysadmin@localhost:/etc/perl$ cd Net
sysadmin@localhost:/etc/perl/Net$
```

Um caminho relativo fornece direções usando sua **localização atual** como um ponto de referência. Lembre-se de que isso é diferente dos caminhos absolutos, que sempre exigem que você use o **diretório raiz** como um ponto de referência.

Há uma técnica de caminho relativo acessível que você pode usar para subir um nível na estrutura de diretórios: o diretório `..`. Independentemente do diretório em que você está, sempre representa um diretório maior do que o diretório atual (com exceção de quando você está no diretório `/`):

```
sysadmin@localhost:/etc/perl/Net$ pwd
/etc/perl/Net
sysadmin@localhost:/etc/perl/Net$ cd ..
sysadmin@localhost:/etc/perl$ pwd
/etc/perl
sysadmin@localhost:/etc/perl$
```

Às vezes, o uso de nomes de caminhos relativos é uma escolha melhor do que nomes de caminhos absolutos, no entanto, isso nem sempre é o caso. Considere se você estava no diretório `/etc/perl/Net` e, em seguida, queria ir para o diretório `/usr/share/doc`. Usando um nome de caminho absoluto, você executaria o comando `cd /usr/share/doc`. Usando nomes de caminhos relativos, você executaria o comando `cd ../../../usr/share/doc`:

```
sysadmin@localhost:/etc/perl/Net$ cd
sysadmin@localhost:~$ cd /etc/perl/Net
sysadmin@localhost:/etc/perl/Net$ cd /../../../usr/share/doc
sysadmin@localhost:/usr/share/doc$ pwd
/usr/share/doc
sysadmin@localhost:/usr/share/doc$
```

Nota: Caminhos relativos e absolutos não são apenas para o comando `cd`. Sempre que você especificar um arquivo ou um diretório, poderá usar caminhos relativos ou absolutos.

Enquanto o ponto duplo (`..`) é usado para se referir ao diretório acima do diretório atual, o ponto único (`.`) é usado para se referir ao diretório atual. Seria inútil para um administrador passar para o diretório atual digitando `cd .` (embora realmente funcione). É mais útil se referir a um item no diretório atual usando a notação `./` . Por exemplo:

```
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$ cd ./Downloads/
sysadmin@localhost:~/Downloads$ pwd
/home/sysadmin/Downloads
sysadmin@localhost:~/Downloads$ cd ..
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

Nota: Esse uso do ponto único (`.`) como ponto de referência não deve ser confundido com o uso no início de um nome de arquivo. Leia mais sobre arquivos ocultos na Seção 6.4.2.

6.3 Listando arquivos num Diretório

Agora que você é capaz de mover de um diretório para outro, convém começar a exibir o conteúdo desses diretórios. O comando `ls` (`ls` é a abreviatura de list) pode ser usado para exibir o conteúdo de um diretório, bem como informações detalhadas sobre os arquivos que estão dentro de um diretório.

Por si só, o comando `ls` listará os arquivos no diretório atual:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$
```

6.3.1 Listagem de cores

Existem muitos tipos diferentes de arquivos no Linux. Ao aprender mais sobre o Linux, você descobrirá muitos desses tipos. A seguir, um breve resumo de alguns dos tipos de arquivos mais comuns:

Tipo	Descrição
arquivo simples	Um arquivo que não é um tipo de arquivo especial; também chamado de arquivo normal
diretório	Um arquivo de diretório (contém outros arquivos)
Executável	Um arquivo que pode ser executado como um programa
Link simbólicos	Um arquivo que aponta para outro arquivo

Em muitas distribuições do Linux, contas de usuário regulares são modificadas para que o comando `ls` exiba nomes de arquivos codificados por cores por tipo de arquivo. Por exemplo, os diretórios podem ser exibidos em azul, os arquivos executáveis podem ser exibidos em verde e os links simbólicos podem ser exibidos em ciano (azul claro).

Este não é um comportamento normal para o comando `ls`, mas sim algo que acontece quando você usa a opção `--color` para o comando `ls`. A razão pela qual o `ls` parece executar automaticamente essa coloração é que existe um alias para o comando `ls`, então ele é executado com a opção `--color`:

```
sysadmin@localhost:~$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
sysadmin@localhost:~$
```

Como você pode ver na saída acima, quando o comando ls é executado, ele realmente executa o comando `ls --color=auto`.

Em alguns casos, você pode não querer ver todas as cores (elas podem ser um pouco perturbadoras às vezes). Para evitar o uso do alias, coloque um caractere de barra invertida \ na frente do seu comando:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$ \ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$
```

6.3.2 Listando arquivos ocultos

Quando você usa o comando `ls` para exibir o conteúdo de um diretório, nem todos os arquivos são mostrados automaticamente. O comando `ls` não exibe arquivos ocultos por padrão. Um arquivo oculto é qualquer arquivo (ou diretório) que começa com um caractere ponto `.`.

Para exibir todos os arquivos, incluindo arquivos ocultos, use a opção `-a` para o comando `ls`:

```
sysadmin@localhost:~$ ls -a
.              .bashrc    .selected_editor  Downloads  Public
..             .cache     Desktop           Music      Templates
.bash_logout   .profile   Documents         Pictures   Videos
```

Por que os arquivos estão ocultos em primeiro lugar? A maioria dos arquivos ocultos são arquivos de personalização, criados para personalizar o funcionamento do Linux, do shell ou dos programas. Por exemplo, o arquivo `.bashrc` no diretório home personaliza os recursos do shell, como criar ou modificar variáveis e aliases.

Esses arquivos de personalização não são aqueles com os quais você trabalha regularmente. Há também muitos deles, como você pode ver, e exibi-los dificultará a localização dos arquivos com os quais você trabalha regularmente. Então, o fato de eles estarem escondidos é para o seu benefício.

6.3.3 Listagem de exibição longa

Há informações sobre cada arquivo, chamados metadados, que às vezes são úteis para serem exibidos. Isso pode incluir quem possui um arquivo, o tamanho de um arquivo e a última vez que o conteúdo de um arquivo foi modificado. Você pode exibir essas informações usando a opção `-l` para o comando `ls`. Abaixo, uma listagem do diretório `/var/log` é usada como exemplo, já que fornece uma variedade de saída:

```
sysadmin@localhost:~$ ls -l /var/log/
total 832
-rw-r--r-- 1 root   root  17869 Mar 14 17:48 alternatives.log
drwxr-x--- 2 root   adm    4096 Mar 14 17:48 apache2
drwxr-xr-x 2 root   root   4096 Mar 14 17:45 apt
-rw-r----- 1 syslog adm     380 Jul 28 03:45 auth.log
-rw-r--r-- 5 root   root  47816 Mar  2 23:10 bootstrap.log
-rw-rw---- 5 root   utmp      0 Mar  2 23:10 btmp
-rw-r----- 1 syslog adm     324 Jul 28 03:45 cron.log
-rw-r----- 1 root   adm   85083 Mar 14 17:48 dmesg
-rw-r--r-- 1 root   root 315196 Mar 14 17:48 dpkg.log
-rw-r--r-- 1 root   root  32064 Mar 14 17:48 faillog
drwxr-xr-x 2 root   root   4096 Jun 30 06:53 fsck
-rw-r----- 1 syslog adm     106 Jul 28 03:45 kern.log
-rw-rw-r-- 1 root   utmp 292584 Jul 28 03:45 lastlog
-rw-r----- 1 syslog adm   18703 Jul 28 03:46 syslog
drwxr-xr-x 2 root   root   4096 Apr 11  2014 upstart
-rw-rw-r-- 1 root   utmp    384 Jul 28 03:45 wtmp
```

Na saída acima, cada linha descreve metadados sobre um único arquivo. A seguir, descrevemos cada um dos campos de dados que você verá na saída do comando `ls -l`:

In the output above, each line describes metadata about a single file. The following describes each of the fields of data that you will see in the output of the `ls -l` command:

Tipo de arquivo

```
-rw-r--r-- 1 root   root  17869 Mar 14 17:48 alternatives.log
drwxr-x--- 2 root   adm    4096 Mar 14 17:48 apache2
```

O primeiro caractere de cada linha indica o tipo de arquivo. Os tipos de arquivos são:

Símbolo	Tipo Arquivo	Descrição
d	diretório	Um arquivo usado para armazenar outros arquivos.
-	arquivo regular	Inclui arquivos legíveis, arquivos de imagens, arquivos binários e arquivos compactados.
l	link simbólico	Aponta para outro arquivo.
s	Socket	Permite a comunicação entre processos.
p	Pipe	Permite a comunicação entre processos.
b	Arquivo de dispositivo – orientado a bloco	Usado para se comunicar com o hardware. Operações de entrada e saída realizadas byte a byte de modo sequencial.
c	Arquivo de dispositivo – orientado a caractere	Usado para se comunicar com o hardware. Operações de entrada e saída realizadas em blocos de modo aleatório.

Permissões

```
drwxr-xr-x 1 root root        0 Apr 11 21:58 upstart
```

Os próximos dez caracteres demonstrarão as permissões do arquivo. Permissões indicam como certos usuários podem acessar um arquivo. As permissões serão abordadas em detalhes em um capítulo posterior.

Contagem de Hard Link

```
-rw-r----- 1 syslog adm 23621 Aug 23 15:17 auth.log
```

Esse número indica quantos links físicos apontam para esse arquivo.

Usuário Proprietário

```
-rw-r----- 1 syslog adm 416 Aug 22 15:43 kern.log
```

Todo arquivo é de propriedade de uma conta de usuário. Isso é importante porque o proprietário tem o direito de definir permissões em um arquivo. As permissões serão abordadas em detalhes em um capítulo posterior.

Grupo Proprietário

```
-rw-rw-r-- 1 root utmp 292584 Aug 20 18:44 lastlog
```

Indica qual grupo possui este arquivo, o que é importante porque qualquer membro desse grupo tem um conjunto de permissões no arquivo. As permissões serão abordadas em detalhes em um capítulo posterior.

Tamando do Arquivo

```
-rw-r----- 1 syslog adm 1087150 Aug 23 15:17 syslog.1
```

O tamanho do arquivo em bytes.

Nota: Para diretórios, esse valor não descreve o tamanho total do diretório, mas sim quantos bytes estão reservados para rastrear os nomes de arquivos no diretório. Em outras palavras, ignore este campo para diretórios.

Timestamp

```
drwxr-xr-x 1 root root 32 Jul 17 03:36 fsck
```

Isso indica a hora em que o conteúdo do arquivo foi modificado pela última vez. Para diretórios, esse registro de data e hora indica a última vez em que um arquivo foi incluído ou excluído do diretório.

Nome do arquivo

```
-rw-r--r-- 1 root root 47816 Jul 17 03:36 bootstrap.log
```

O campo final contém o nome do arquivo ou diretório.

Considere isto

No caso de links simbólicos, o nome do link será exibido junto com uma seta e o nome do caminho do arquivo original.

```
lrwxrwxrwx. 1 root root 22 Nov 6 2012 /etc/grub.conf -> ../boot/grub/
grub.conf
```

Os links simbólicos serão abordados em detalhes em um capítulo posterior.

6.3.3.1 Tamanho legível para humanos

Quando você exibe tamanhos de arquivo com a opção `-l` para o comando `ls`, você obtem tamanhos de arquivo em bytes. Para arquivos de texto, um byte é 1 caractere.

Para arquivos menores, os tamanhos de bytes estão bem. No entanto, para arquivos maiores, é difícil compreender o tamanho do arquivo. Por exemplo, considere a saída do seguinte comando:

```
sysadmin@localhost:~$ ls -l /usr/bin/omshell
-rwxr-xr-c 1 root root 1561400 Oct 9 2012 /usr/bin/omshell


sysadmin@localhost:~$
```

Como você pode ver, o tamanho do arquivo é difícil de determinar em bytes. É 1561400 um arquivo grande ou pequeno? Parece bastante grande, mas é difícil determinar o uso de bytes.

Pense nisso desta maneira: se alguém lhe desse a distância entre Boston e Nova York usando polegadas, esse valor seria essencialmente sem sentido porque, para uma distância como essa, você pensa em termos de quilometros.

Seria melhor se o tamanho do arquivo fosse apresentado em um tamanho mais legível, como megabytes ou gigabytes. Para isso, adicione a opção `-h` ao comando `ls`:

```
sysadmin@localhost:~$ ls -lh /usr/bin/omshell
-rwxr-xr-c 1 root root 1.5M Oct 9 2012 /usr/bin/omshell
sysadmin@localhost:~$
```

Importante: A opção `-h` deve ser usada com a opção `-l`.

6.3.4 Listando Diretórios

Quando o comando `ls -d` é usado, ele se refere ao diretório atual e não ao conteúdo dentro dele. Sem quaisquer outras opções, é bastante insignificante, embora seja importante observar que o diretório atual é sempre referido com um único ponto (`.`):

```
sysadmin@localhost:~$ ls -d
.
```

Para usar o comando `ls -d` de uma maneira mais significativa, é necessário adicionar a opção `-l`. Nesse caso, observe que o primeiro comando lista os detalhes do conteúdo no diretório `/home/sysadmin`, enquanto o segundo comando lista o próprio diretório `/home/sysadmin`.

```
sysadmin@localhost:~$ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Desktop
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Documents
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Downloads

drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Music
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Pictures
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Public
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Templates
drwxr-xr-x 1 sysadmin sysadmin   0 Apr 15  2015 Videos
drwxr-xr-x 1 sysadmin sysadmin 420 Apr 15  2015 test
sysadmin@localhost:~$ ls -ld
drwxr-xr-x 1 sysadmin sysadmin 224 Nov  7 17:07 .
sysadmin@localhost:~$
```

Observe o período único no final da segunda listagem longa. Isso indica que o diretório atual está sendo listado e não o conteúdo.

6.3.5 Listagem recursiva

Haverá momentos em que você deseja exibir todos os arquivos em um diretório, bem como todos os arquivos em todos os subdiretórios em um diretório. Isso é chamado de *listagem recursiva*.

Para executar uma listagem recursiva, use a opção `-R` para o comando `ls`:

Nota: A saída mostrada abaixo irá variar dos resultados que você verá se executar o comando no ambiente de máquina virtual deste curso.

```
sysadmin@localhost:~$ ls -R /etc/ppp

/etc/ppp:

chap-secrets    ip-down.ipv6to4    ip-up.ipv6to4    ipv6-up    pap-secr
ets

ip-down         ip-up              ipv6-down        options    peers


/etc/ppp/peers:

sysadmin@localhost:~$
```

Note que no exemplo anterior, os arquivos no diretório `/etc/ppp` foram listados primeiro. Depois disso, os arquivos no diretório `/etc/ppp/peers` foram listados (não havia arquivos neste caso, mas se algum arquivo estivesse nesse diretório, eles teriam sido exibidos).

Tenha cuidado com esta opção; por exemplo, executar o comando `ls -R /` listaria todos os arquivos no sistema de arquivos, incluindo todos os arquivos em qualquer dispositivo USB conectado e DVD no sistema. Limite o uso da opção `-R` para estruturas de diretório menores.

6.3.6 Ordenar uma listagem

Por padrão, o comando `ls` classifica os arquivos em ordem alfabética pelo nome do arquivo. Às vezes, pode ser útil classificar arquivos usando critérios diferentes.

Para classificar arquivos por tamanho, podemos usar a opção `-S`. Observe a diferença na saída dos dois comandos a seguir:

```
sysadmin@localhost:~$ ls /etc/ssh moduli
ssh_host_dsa_key.pub    ssh_host_rsa_key     sshd_confi
ssh_config         ssh_host_ecdsa_key      ssh_host_rsa_key.pub
ssh_host_dsa_key   ssh_host_ecdsa_key.pub  ssh_import_id
sysadmin@localhost:~$ ls -S /etc/ssh moduli
ssh_host_dsa_key       ssh_host_ecdsa_key
sshd_config        ssh_host_dsa_key.pub   ssh_host_ecdsa_key.pub
ssh_host_rsa_key   ssh_host_rsa_key.pub
ssh_config         ssh_import_id
sysadmin@localhost:~$
```

Os mesmos arquivos e diretórios são listados, mas em uma ordem diferente. Enquanto a opção `-S` funciona por si só, você não pode realmente dizer que a saída é classificada por tamanho, por isso é mais útil quando usado com a opção `-l`. O comando a seguir listará os arquivos do maior para o menor e exibirá o tamanho real do arquivo.

```
sysadmin@localhost:~$ ls -lS /etc/ssh
total 160
-rw-r--r-- 1 root root 125749 Apr 29  2014 moduli
-rw-r--r-- 1 root root   2489 Jan 29  2015 sshd_config
-rw------- 1 root root   1675 Jan 29  2015 ssh_host_rsa_key
-rw-r--r-- 1 root root   1669 Apr 29  2014 ssh_config
-rw------- 1 root root    668 Jan 29  2015 ssh_host_dsa_key
-rw-r--r-- 1 root root    607 Jan 29  2015 ssh_host_dsa_key.pub
-rw-r--r-- 1 root root    399 Jan 29  2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root    302 Jan 10  2011 ssh_import_id
-rw------- 1 root root    227 Jan 29  2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    179 Jan 29  2015 ssh_host_ecdsa_key.pub
sysadmin@localhost:~$
```

Também pode ser útil usar a opção `-h` para exibir tamanhos de arquivos legíveis por humanos:

```
sysadmin@localhost:~$ ls -lSh /etc/ssh
total 160K
-rw-r--r-- 1 root root 123K Apr 29  2014 moduli
-rw-r--r-- 1 root root 2.5K Jan 29  2015 sshd_config
-rw------- 1 root root 1.7K Jan 29  2015 ssh_host_rsa_key
-rw-r--r-- 1 root root 1.7K Apr 29  2014 ssh_config
-rw------- 1 root root  668 Jan 29  2015 ssh_host_dsa_key
-rw-r--r-- 1 root root  607 Jan 29  2015 ssh_host_dsa_key.pub
-rw-r--r-- 1 root root  399 Jan 29  2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root  302 Jan 10  2011 ssh_import_id
-rw------- 1 root root  227 Jan 29  2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root  179 Jan 29  2015 ssh_host_ecdsa_key.pub
sysadmin@localhost:~$
```

Também é possível classificar arquivos com base na hora em que foram modificados. Você pode fazer isso usando a opção `-t`.

A opção `-t` listará os arquivos modificados mais recentemente primeiro. Esta opção pode ser usada sozinha, mas novamente é mais útil quando emparelhada com a opção `-l`:

```
sysadmin@localhost:~$ ls -tl /etc/ssh
total 160
-rw------- 1 root root    668 Jan 29  2015 ssh_host_dsa_key
-rw-r--r-- 1 root root    607 Jan 29  2015 ssh_host_dsa_key.pub
-rw------- 1 root root    227 Jan 29  2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    179 Jan 29  2015 ssh_host_ecdsa_key.pub
-rw------- 1 root root   1675 Jan 29  2015 ssh_host_rsa_key
-rw-r--r-- 1 root root    399 Jan 29  2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root   2489 Jan 29  2015 sshd_config
-rw-r--r-- 1 root root 125749 Apr 29  2014 moduli
-rw-r--r-- 1 root root   1669 Apr 29  2014 ssh_config
-rw-r--r-- 1 root root    302 Jan 10  2011 ssh_import_id
sysadmin@localhost:~$
```

É importante lembrar que a data modificada nos diretórios representa a última vez que um arquivo foi adicionado ou removido do diretório.

Se os arquivos em um diretório foram modificados muitos dias ou meses atrás, pode ser mais difícil dizer exatamente quando eles foram modificados, pois somente a data é

fornecida para arquivos mais antigos. Para informações detalhadas sobre o tempo de modificação, você pode usar a opção `--full-time` para exibir o registro de data e hora completo (incluindo horas, segundos, minutos ...):

```
sysadmin@localhost:~$ ls -t --full-time /etc/ssh
total 160
-rw------- 1 root root    668 2015-01-29 03:17:33.000000000 +0000 ssh_
host_dsa_key
-rw-r--r-- 1 root root    607 2015-01-29 03:17:33.000000000 +0000 ssh_
host_dsa_key.pub
-rw------- 1 root root    227 2015-01-29 03:17:33.000000000 +0000 ssh_
host_ecdsa_key
-rw-r--r-- 1 root root    179 2015-01-29 03:17:33.000000000 +0000 ssh_
host_ecdsa_key.pub
-rw------- 1 root root   1675 2015-01-29 03:17:33.000000000 +0000 ssh_
host_rsa_key
-rw-r--r-- 1 root root    399 2015-01-29 03:17:33.000000000 +0000 ssh_
host_rsa_key.pub
-rw-r--r-- 1 root root   2489 2015-01-29 03:17:33.000000000 +0000 sshd
_config
-rw-r--r-- 1 root root 125749 2014-04-29 23:58:51.000000000 +0000 modu
li
-rw-r--r-- 1 root root   1669 2014-04-29 23:58:51.000000000 +0000 ssh_
config
-rw-r--r-- 1 root root    302 2011-01-10 18:48:29.000000000 +0000 ssh_
import_id
sysadmin@localhost:~$
```

A opção `--full-time` assumirá a opção `-l` automaticamente.

É possível executar uma ordenação reversa com as opções `-S` ou `-t` usando a opção `-r`. O comando a seguir classificará os arquivos por tamanho, do menor para o maior:

```
sysadmin@localhost:~$ ls -lrS /etc/ssh
total 160
-rw-r--r-- 1 root root    179 Jan 29  2015 ssh_host_ecdsa_key.pub
-rw------- 1 root root    227 Jan 29  2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    302 Jan 10  2011 ssh_import_id
-rw-r--r-- 1 root root    399 Jan 29  2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root    607 Jan 29  2015 ssh_host_dsa_key.pub
-rw------- 1 root root    668 Jan 29  2015 ssh_host_dsa_key
-rw-r--r-- 1 root root   1669 Apr 29  2014 ssh_config
-rw------- 1 root root   1675 Jan 29  2015 ssh_host_rsa_key
-rw-r--r-- 1 root root   2489 Jan 29  2015 sshd_config

-rw-r--r-- 1 root root 125749 Apr 29  2014 moduli
sysadmin@localhost:~$
```

O comando a seguir listará os arquivos por data de modificação, da mais antiga para a mais recente:

```
sysadmin@localhost:~$ ls -lrt /etc/ssh
total 160
-rw-r--r-- 1 root root    302 Jan 10  2011 ssh_import_id
-rw-r--r-- 1 root root   1669 Apr 29  2014 ssh_config
-rw-r--r-- 1 root root 125749 Apr 29  2014 moduli
-rw-r--r-- 1 root root   2489 Jan 29  2015 sshd_config
-rw-r--r-- 1 root root    399 Jan 29  2015 ssh_host_rsa_key.pub
-rw------- 1 root root   1675 Jan 29  2015 ssh_host_rsa_key
-rw-r--r-- 1 root root    179 Jan 29  2015 ssh_host_ecdsa_key.pub
-rw------- 1 root root    227 Jan 29  2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    607 Jan 29  2015 ssh_host_dsa_key.pub
-rw------- 1 root root    668 Jan 29  2015 ssh_host_dsa_key
sysadmin@localhost:~$
```

6.3.7 Listagem com Globs

Em um capítulo anterior, discutimos o uso de globs de arquivos para corresponder nomes de arquivos usando caracteres curinga. Por exemplo, demonstramos que você pode listar todos os arquivos no diretório `/etc` que começam com a letra `e` com o seguinte comando:

```
sysadmin@localhost:~$ echo /etc/e*
/etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/export
s
sysadmin@localhost:~$
```

Agora que você sabe que o comando `ls` é normalmente usado para listar arquivos em um diretório, usar o comando `echo` pode parecer uma escolha estranha. No entanto, há algo sobre o comando `ls` que pode ter causado confusão enquanto estávamos discutindo globs. Esse "recurso" também pode causar problemas ao tentar listar arquivos usando padrões glob.

Tenha em mente que é o shell, não o comando `echo` ou `ls`, que expande o padrão glob nos nomes de arquivo correspondentes. Em outras palavras, quando você digita o comando `echo /etc/e*`, o que o shell fez antes de executar o comando echo foi substituir `e*` por todos os arquivos e diretórios dentro do diretório `/etc` que correspondem ao padrão.

Então, se você fosse executar o comando `ls /etc/e*`, o que o shell realmente executaria seria:

```
ls /etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/exp
orts
```

Quando o comando `ls` vê vários argumentos, ele executa uma operação de lista em cada item separadamente. Em outras palavras, o comando `ls /etc/encript.cfg /etc/environment` é essencialmente o mesmo que `ls /etc/encript.cfg; ls /etc/environment`.

Agora considere o que acontece quando você executa o comando `ls` em um arquivo, como `encript.cfg`:

```
sysadmin@localhost:~$ ls /etc/enscript.cfg


/etc/enscript.cfg
sysadmin@localhost:~$
```

Como você pode ver, executar o comando `ls` em um único arquivo resulta no nome do arquivo que está sendo impresso. Normalmente, isso é útil se você quiser ver detalhes sobre um arquivo específico usando a opção `-l` para o comando `ls`:

```
sysadmin@localhost:~$ ls -l /etc/enscript.cfg
-r--r--r--. 1 root root 4843 Nov 11 2010 /etc/enscript.cfg
sysadmin@localhost:~$
```

No entanto, e se o comando `ls` receber um nome de diretório como argumento? Nesse caso, a saída do comando é diferente de se o argumento fosse um nome de arquivo:

```
sysadmin@localhost:~$ ls /etc/event.d
ck-log-system-restart  ck-log-system-start  ck-log-system-stop
sysadmin@localhost:~$
```

Se você der um nome de diretório como um argumento para o comando `ls`, o comando exibirá o ***conteúdo*** do diretório (os nomes dos arquivos no diretório), e não apenas fornecerá o nome do diretório. Os nomes de arquivos que você vê no exemplo acima são os nomes dos arquivos no diretório `/etc/event.d`.

Por que isso é um problema ao usar globs? Considere a seguinte saída:

```
sysadmin@localhost:~$ ls /etc/e*
/etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/export
s
/etc/event.d:
ck-log-system-restart  ck-log-system-start  ck-log-system-stop
sysadmin@localhost:~$
```

Como você pode ver, quando o comando `ls` vê um nome de arquivo como um argumento, ele simplesmente exibe o nome do arquivo. No entanto, para qualquer diretório, ele exibirá o conteúdo do diretório, não apenas o nome do diretório.

Isso se torna ainda mais confuso em uma situação como a seguinte:

```
sysadmin@localhost:~$ ls /etc/ev*
ck-log-system-restart  ck-log-system-start  ck-log-system-stop
sysadmin@localhost:~$
```

No exemplo anterior, parece que o comando `ls` está simplesmente errado. Mas o que realmente aconteceu é que a única coisa que corresponde ao glob `/etc/ev*` é o diretório `/etc/event.d`. Portanto, o comando `ls` exibiu apenas os arquivos nesse diretório!

Há uma solução simples para esse problema: quando você usa argumentos glob com o comando `ls`, sempre use a opção `-d`. Quando você usa a opção `-d`, o comando `ls` não exibe o conteúdo de um diretório, mas sim o nome do diretório:

```
sysadmin@localhost:~$ ls -d /etc/e*
/etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/export
s
sysadmin@localhost:~$
```

6.4 Copiando Arquivos

O comando `cp` é usado para copiar arquivos. Isso requer que você especifique uma origem e um destino. A estrutura do comando é a seguinte:

```
cp [origem] [destino]
```

A origem é o arquivo que você deseja copiar. O destino é onde você deseja que a cópia seja localizada. Quando bem sucedido, o comando `cp` não terá qualquer saída (nenhuma notícia é algo bom). O seguinte comando irá copiar o arquivo `/etc/hosts` para o seu diretório home:

```
sysadmin@localhost:~$ cp /etc/hosts ~
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  hosts
Documents  Music      Public    Videos
sysadmin@localhost:~$
```

Lembre-se: O caractere `~` representa seu diretório home.

6.4.1 Modo Verbose

A opção `-v` fará com que o comando `cp` produza saída se for bem-sucedido. A opção `-v` significa verbose (detalhado):

```
sysadmin@localhost:~$ cp -v /etc/hosts ~
`/etc/hosts' -> `/home/sysadmin/hosts'
sysadmin@localhost:~$
```

Quando o destino é um diretório, o novo arquivo resultante terá o mesmo nome do arquivo original. Se você quiser que o novo arquivo tenha um nome diferente, forneça o novo nome como parte do destino:

```
sysadmin@localhost:~$ cp /etc/hosts ~/hosts.copy
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  hosts
Documents  Music      Public    Videos     hosts.copy
sysadmin@localhost:~$
```

6.4.2 Evite Sobrescrever Dados

O comando `cp` pode ser destrutivo para os dados existentes, se o arquivo de destino já existir. No caso em que o arquivo de destino existe, o comando `cp` sobrescreverá o conteúdo do arquivo existente com o conteúdo do arquivo de origem. Para ilustrar esse problema em potencial, primeiro é criado um novo arquivo no diretório home do usuário `sysadmin` copiando um arquivo existente:

```
sysadmin@localhost:~$ cp /etc/skel/.bash_logout ~/example.txt
sysadmin@localhost:~$
```

Visualize a saída do comando `ls` para ver o arquivo e visualizar o conteúdo do arquivo usando o comando `more`:

```
sysadmin@localhost:~$ cp /etc/skel/.bash_logout ~/example.txt
sysadmin@localhost:~$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
sysadmin@localhost:~$ more example.txt
# ~/.bash_logout: executed by bash(1) when login shell exits.

sysadmin@localhost:~$ cp -i /etc/hosts ~/example.txt
cp: overwrite `/home/sysadmin/example.txt'? n
sysadmin@localhost:~$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
sysadmin@localhost:~$ more example.txt
# ~/.bash_logout: executed by bash(1) when login shell exits.

sysadmin@localhost:~$
```

No próximo exemplo, você verá que o comando `cp` destrói o conteúdo original do arquivo `example.txt`. Observe que após o comando `cp` ser concluído, o tamanho do arquivo é diferente (158 bytes em vez de 18) do original e os conteúdos também são diferentes:

```
sysadmin@localhost:~$ cp /etc/hosts ~/example.txt
sysadmin@localhost:~$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 158 Sep 21 14:11 example.txt
sysadmin@localhost:~$ cat example.txt
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.local
domain4
::1         localhost localhost.localdomain localhost6 localhost6.local
domain6
sysadmin@localhost:~$
```

Existem duas opções que podem ser usadas para proteger contra substituições acidentais. Com a opção `-i` (interativa), o `cp` perguntará antes de sobrescrever um arquivo. O exemplo a seguir demonstrará essa opção, primeiro restaurando o conteúdo do arquivo original:

```
sysadmin@localhost:~$ cp /etc/skel/.bash_logout ~/example.txt
sysadmin@localhost:~$ ls -l example.txt
-rw-r--r-- 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
sysadmin@localhost:~$ more example.txt
# ~/.bash_logout: executed by bash(1) when login shell exits.

sysadmin@localhost:~$ cp -i /etc/hosts ~/example.txt
cp: overwrite `/home/sysadmin/example.txt'? n
sysadmin@localhost:~$ ls -l example.txt
-rw-r--r-- 1 sysadmin sysadmin 18 Sep  21 15:56 example.txt
sysadmin@localhost:~$ more example.txt
# ~/.bash_logout: executed by bash(1) when login shell exits.

sysadmin@localhost:~$
```

Observe que, como o valor de `n` (no) foi fornecido quando solicitado a sobrescrever o arquivo, nenhuma alteração foi feita no arquivo. Se um valor de `y` (sim) foi dado, então o processo de cópia teria ocorrido.

A opção `-i` requer que você responda `y` ou `n` para cada cópia que pode acabar substituindo o conteúdo de um arquivo existente. Isso pode ser entediante quando um grupo de substituições pode ocorrer, como o exemplo demonstrado abaixo:

```
sysadmin@localhost:~$ cp -i /etc/skel/.* ~
cp: omitting directory `/etc/skel/.'
cp: omitting directory `/etc/skel/..'
cp: overwrite `/home/sysadmin/.bash_logout'? n
cp: overwrite `/home/sysadmin/.bashrc'? n
cp: overwrite `/home/sysadmin/.profile'? n
cp: overwrite `/home/sysadmin/.selected_editor'? n
sysadmin@localhost:~$
```

Como você pode ver no exemplo acima, o comando `cp` tentou substituir quatro arquivos existentes, forçando o usuário a responder três prompts. Se essa situação acontecesse por 100 arquivos, poderia se tornar muito irritante, muito rapidamente.

Se você quiser responder automaticamente **n** a cada prompt, use a opção `-n`. Essencialmente significa "não reescrever".

6.4.3 Copiando Diretórios

Em um exemplo anterior, mensagens de erro foram fornecidas quando o comando `cp` tentou copiar diretórios:

```
sysadmin@localhost:~$ cp -i /etc/skel/.* ~
cp: omitting directory `/etc/skel/.'


cp: omitting directory `/etc/skel/..'
cp: overwrite `/home/sysadmin/.bash_logout'? n
cp: overwrite `/home/sysadmin/.bashrc'? n
cp: overwrite `/home/sysadmin/.profile'? n
cp: overwrite `/home/sysadmin/.selected_editor'? n
sysadmin@localhost:~$
```

Onde a saída diz `...omitting directory...`, o comando `cp` está dizendo que não pode copiar este item porque o comando não copia diretórios por padrão. No entanto, a opção `-r` para o comando `cp` fará com que copie os arquivos e diretórios.

Tenha cuidado com esta opção: toda a estrutura do diretório será copiada. Isso pode resultar na cópia de muitos arquivos e diretórios!

6.5 Movendo Arquivos

Para mover um arquivo, use o comando `mv`. A sintaxe para o comando `mv` é muito parecida com o comando `cp`:

```
mv [origem] [destination]
```

No exemplo a seguir, o arquivo `hosts` gerado anteriormente é movido do diretório atual para o diretório `Videos`:

```
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  example.txt  hosts.copy
Documents  Music      Public    Videos     hosts
sysadmin@localhost:~$ mv hosts Videos
sysadmin@localhost:~$ ls

Desktop    Downloads  Pictures  Templates  example.txt
Documents  Music      Public    Videos     hosts.copy
sysadmin@localhost:~$ ls Videos
hosts
sysadmin@localhost:~$
```

Quando um arquivo é movido, o arquivo é removido do local original e colocado em um novo local. Isso pode ser um pouco complicado no Linux, porque os usuários precisam de permissões específicas para remover arquivos de um diretório. Se você não tiver as permissões corretas, receberá uma mensagem de erro `Permission denied` (Permissão negada):

```
sysadmin@localhost:~$ mv /etc/hosts .
mv: cannot move `/etc/hosts' to `./hosts': Permission denied
sysadmin@localhost:~$
```

Uma discussão detalhada de permissões é fornecida em um capítulo posterior.

6.6 Movendo arquivos durante a renomeação

Se o destino para o comando `mv` for um diretório, o arquivo será movido para o diretório especificado. O nome do arquivo será alterado apenas se um nome de arquivo de destino também for especificado.

Se um diretório de destino não for especificado, o arquivo será renomeado usando o nome do arquivo de destino e permanecerá no diretório de origem.

```
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  example.txt
Documents  Music      Public    Videos
sysadmin@localhost:~$ mv example.txt Videos/newexample.txt
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates
Documents  Music      Public    Videos
sysadmin@localhost:~$ ls Videos
hosts  newexample.txt


sysadmin@localhost:~$
```

6.6.1 Renomeando Arquivos

O comando `mv` não é usado apenas para mover um arquivo, mas também para renomear um arquivo. Por exemplo, os seguintes comandos renomearão o arquivo `newexample.txt` para `myexample.txt`:

```
sysadmin@localhost:~$ cd Videos
sysadmin@localhost:~/Videos$ ls
hosts  newexample.txt
sysadmin@localhost:~/Videos$ mv newexample.txt myexample.txt
sysadmin@localhost:~/Videos$ ls
hosts  myexample.txt
sysadmin@localhost:~/Videos$
```

Pense no exemplo `mv` anterior para "mover o arquivo `newexample.txt` do diretório atual de volta para o diretório atual e dar ao novo arquivo o nome `myexample.txt`.

6.6.2 Opções mv adicionais

Como o comando `cp`, o comando `mv` fornece as seguintes opções:

Opção	Significado
`-i`	Movimento interativo: pergunte se um arquivo deve ser sobrescrito.
`-n`	Não sobrescreve o conteúdo de um arquivo de destino
`-v`	Verbose: mostra o resultado do movimento

Importante: Não há opção `-r`, pois o comando `mv` irá, por padrão, mover os diretórios.

6.7 Criando Arquivos

Existem várias maneiras de criar um novo arquivo, incluindo o uso de um programa projetado para editar um arquivo (um editor de texto). Em um capítulo posterior, os editores de texto serão abordados.

Há também uma maneira de simplesmente criar um arquivo que possa ser preenchido com dados posteriormente. Esse é um recurso útil, pois, para alguns recursos do sistema operacional, a própria existência de um arquivo pode alterar o funcionamento de um comando ou serviço. Também é útil criar um arquivo como "espaço reservado" para lembrá-lo de criar o conteúdo do arquivo posteriormente.

Para criar um arquivo vazio, use o comando `touch` como demonstrado abaixo:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$ touch sample
sysadmin@localhost:~$ ls -l sample


-rw-rw-r-- 1 sysadmin sysadmin 0 Nov  9 16:48 sample
sysadmin@localhost:~$
```

Observe que o tamanho do novo arquivo é de 0 bytes. Como mencionado anteriormente, o comando `touch` não coloca nenhum dado dentro do novo arquivo.

6.8 Removendo Arquivos

Para excluir um arquivo, use o comando `rm`:

```
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  sample
Documents  Music      Public    Videos
sysadmin@localhost:~$ rm sample
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$
```

Observe que o arquivo foi excluído sem perguntas. Isso pode causar problemas ao excluir vários arquivos usando caracteres glob, por exemplo: `rm *.txt`. Como esses arquivos são excluídos sem questionamentos, um usuário pode acabar excluindo arquivos que não deveriam ser excluídos.

Além disso, os arquivos são excluídos permanentemente. Não há nenhum comando para desfazer a exclusão de um arquivo e nenhuma "lixeira" da qual recuperar arquivos excluídos. Como precaução, os usuários devem usar a opção `-i` ao excluir vários arquivos:

```
sysadmin@localhost:~$ touch sample.txt example.txt test.txt
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  example.txt  test.txt
Documents  Music      Public    Videos     sample.txt
sysadmin@localhost:~$ rm -i *.txt
rm: remove regular empty file `example.txt'? y
rm: remove regular empty file `sample.txt'? n
rm: remove regular empty file `test.txt'? y
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  sample.txt
Documents  Music      Public    Videos
sysadmin@localhost:~$
```

6.9 Removendo Diretórios

Você pode excluir diretórios usando o comando `rm`. No entanto, o uso padrão (sem opções) do comando `rm` falhará ao excluir um diretório:

```
sysadmin@localhost:~$ rm Videos
rm: cannot remove `Videos': Is a directory
sysadmin@localhost:~$
```

Se você quiser excluir um diretório, use a opção `-r` para o comando `rm`:

```
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  sample.txt
Documents  Music      Public    Videos
sysadmin@localhost:~$ rm -r Videos
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
sample.txt
sysadmin@localhost:~$
```

Importante: Quando um usuário exclui um diretório, todos os arquivos e subdiretórios são excluídos sem qualquer pergunta interativa. É melhor usar a opção `-i` com o comando `rm`.

Você também pode excluir um diretório com o comando `rmdir`, mas apenas se o diretório estiver vazio.

6.10 Criando Diretórios

Para criar um diretório, use o comando `mkdir`:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
sample.txt
sysadmin@localhost:~$ mkdir test
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates   test
Documents  Music      Public    sample.txt


sysadmin@localhost:~$
```