Linux
Professional
Institute

Tópico 104: Dispositivos, sistemas de arquivos Linux e padrão FHS

104.1 Criar partições e sistemas de arquivos

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 104.1

Peso

2

Áreas chave de conhecimento

- Gerenciar tabela de partição MBR e GPT
- Usar vários comandos mkfs para criar sistemas de arquivos tais como:
  - ext2/ext3/ext4
  - XFS
  - VFAT
  - exFAT
- Conhecimento básico dos recursos do Btrfs, incluindo sistema de arquivos em multidispositivos, compressão e subvolumes.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `fdisk`
- `gdisk`
- `parted`
- `mkfs`
- `mkswap`

104.1 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	104 Dispositivos, sistemas de arquivos do Linux, hierarquia padrão de sistemas de arquivos
Objetivo:	104.1 Criação de partições e sistemas de arquivos
Lição:	1 de 1

Introdução

Em qualquer sistema operacional, um disco precisa ser particionado antes de poder ser usado. Uma partição é um subconjunto lógico do disco físico; as informações sobre as partições são armazenadas em uma tabela de partições. Essa tabela inclui informações sobre o primeiro e o último setores da partição e seu tipo, além de mais detalhes sobre cada partição.

Normalmente, cada partição é vista por um sistema operacional como um "disco" separado, mesmo que todas residam na mesma mídia física. Nos sistemas Windows, elas recebem letras como `C:` (historicamente o disco principal), `D:` e assim por diante. No Linux, cada partição recebe um diretório em `/dev`, como `/dev/sda1` ou `/dev/sda2`.

Nesta lição, você aprenderá a criar, excluir, restaurar e redimensionar partições usando os três utilitários mais comuns (`fdisk`, `gdisk` e `parted`), a criar um sistema de arquivos nelas e a criar e definir uma *partição de troca* ou *arquivo de troca* para ser usado como memória virtual.

NOTE Por razões históricas, nesta lição nos referimos às mídias de armazenamento como "discos", mesmo que os sistemas de armazenamento modernos, como SSDs e armazenamento flash, não contenham mais nenhum "disco".

Entendendo MBR e GPT

Existem duas maneiras principais de armazenar informações sobre partições em discos rígidos. A primeira é o MBR (*Master Boot Record*, ou Registro Mestre de Inicialização) e a segunda é a GPT (*GUID Partition Table*, ou Tabela de Partição GUID).

MBR

Um remanescente dos primeiros dias do MS-DOS (mais especificamente, o PC-DOS 2.0 de 1983) que, por décadas, foi o esquema de particionamento padrão dos PCs. A tabela de partição é armazenada no primeiro setor de um disco, chamado *setor de inicialização*, junto com um carregador de inicialização, que em sistemas Linux geralmente é o bootloader *GRUB*. Mas o MBR tem uma série de limitações que dificultam seu uso em sistemas modernos, como a incapacidade de endereçar discos com mais de 2 TB de tamanho e o limite de apenas 4 partições primárias por disco.

GUID

Um sistema de particionamento que aborda muitas das limitações do MBR. Não há limite prático para o tamanho do disco, e o número máximo de partições é limitado apenas pelo próprio sistema operacional. É mais comumente encontrado em máquinas mais modernas que usam UEFI em vez da antiga BIOS.

Durante as tarefas de administração do sistema, é bastante possível que você encontre ambos os esquemas em uso, por isso é importante saber como usar as ferramentas associadas a cada um para criar, excluir ou modificar partições.

Gerenciando partições MBR com o FDISK

O utilitário padrão para gerenciar partições MBR no Linux é o `fdisk`. Trata-se de um utilitário interativo com menu. Para usá-lo, digite `fdisk` seguido pelo nome do dispositivo correspondente ao disco que deseja editar. Por exemplo, o comando

```
# fdisk /dev/sda
```

serve para editar a tabela de partição do primeiro dispositivo conectado por SATA (`sda`) no sistema. Lembre-se de que é preciso especificar o dispositivo correspondente ao disco físico, não uma de suas partições (como `/dev/sda1`).

NOTE Todas as operações de disco desta lição devem ser realizadas com o usuário `root` (o administrador do sistema), ou com privilégios de root usando `sudo`.

Quando invocado, `fdisk` mostra uma saudação seguida de um aviso e espera pelos seus comandos.

```
# fdisk /dev/sda
Welcome to fdisk (util-linux 2.33.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

O aviso é importante. Você pode criar, editar ou remover partições à vontade, mas *nada* será gravado no disco a menos que você use o comando write (`w`). Assim, você pode "praticar" sem o risco de perder dados, desde que mantenha distância da tecla `w`. Para sair do `fdisk` sem salvar as alterações, use o comando `q`.

NOTE Dito isso, jamais pratique em um disco importante, pois sempre haverá riscos. Use um disco externo sobressalente ou um pendrive.

Imprimindo a tabela de partição atual

O comando `p` é usado para imprimir a tabela de partição atual. A saída é mais ou menos assim:

```
Command (m for help): p
Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors
Disk model: CT120BX500SSD1
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x97f8fef5

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1             4096 226048942 226044847 107.8G 83 Linux
/dev/sda2       226048944 234437550   8388607     4G 82 Linux swap / Solaris
```

Este é o significado de cada coluna:

`Device`

O dispositivo atribuído à partição.

`Boot`

Mostra se a partição é "inicializável" ou não.

`Start`

O setor em que a partição começa.

`End`

O setor em que a partição termina.

`Sectors`

O número total de setores na partição. Deve ser multiplicado pelo tamanho dos setores para se obter o tamanho da partição em bytes.

`Size`

O tamanho da partição em formato "legível por humanos". No exemplo acima, os valores estão em gigabytes.

`Id`

O valor numérico que representa o tipo de partição.

`Type`

A descrição do tipo de partição.

Partições primárias e estendidas

Em um disco MBR, podemos ter 2 tipos principais de partições, *primária* e *estendida*. Como já dissemos, só é possível ter 4 partições primárias no disco e, para que o disco seja "inicializável", a primeira partição deve ser primária.

Uma maneira de contornar essa limitação é criar uma partição estendida que atue como um contêiner para partições *lógicas*. Seria possível ter, por exemplo, uma partição primária, uma partição estendida ocupando o restante do espaço em disco e cinco partições lógicas dentro dela.

Para um sistema operacional como o Linux, as partições primárias e estendidas são tratadas exatamente da mesma maneira, então não há "vantagens" em se usar uma ou outra.

Criando uma partição

Para criar uma partição, use o comando `n`. Por padrão, as partições serão criadas no início do espaço não alocado no disco. Você será questionado sobre o tipo de partição (primária ou estendida), primeiro setor e último setor.

Para o primeiro setor, geralmente podemos aceitar o valor padrão sugerido pelo `fdisk`, a menos que

você precise que uma partição inicie em um setor específico. Em vez de especificar o último setor, dá para especificar um tamanho seguido das letras `K`, `M`, `G`, `T` ou `P` (Kilo, Mega, Giga, Tera ou Peta). Assim, se você quiser criar uma partição de 1 GB, pode especificar `+1G` como `Last sector` e o `fdisk` redimensiona a partição de acordo. Veja este exemplo para a criação de uma partição primária:

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-3903577, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-3903577, default 3903577): +1G
```

Verificando o espaço não alocado

Se você não souber quanto espaço livre resta no disco, pode usar o comando `F` para mostrar o espaço não alocado, desta maneira:

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start     End Sectors  Size
2099200 3903577 1804378  881M
```

Removendo partições

Para remover uma partição, use o comando `d`. O `fdisk` irá pedir o número da partição a remover, *a menos* que haja *apenas uma* partição no disco. Neste caso, essa partição será *selecionada e excluída imediatamente*.

Esteja ciente de que se você excluir uma partição estendida, todas as partições lógicas dentro dela também serão excluídas.

Lacunas

Tenha em mente que, ao criar uma nova partição com `fdisk`, o tamanho máximo será limitado pela quantidade máxima de espaço *contíguo* não alocado no disco. Digamos, por exemplo, que você tenha

o seguinte mapa de partições:

```
Device     Boot   Start     End Sectors  Size Id Type
/dev/sdd1          2048 1050623 1048576  512M 83 Linux
/dev/sdd2       1050624 2099199 1048576  512M 83 Linux
/dev/sdd3       2099200 3147775 1048576  512M 83 Linux
```

Em seguida, você exclui a partição 2 e verifica o espaço livre:

```
Command (m for help): F
Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

  Start     End Sectors  Size
1050624 2099199 1048576  512M
3147776 3903577  755802  369M
```

Somando o tamanho do espaço não alocado, em teoria teríamos 881 MB disponíveis. Mas veja o que acontece quando tentamos criar uma partição de 700 MB:

```
Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2,4, default 2): 2
First sector (1050624-3903577, default 1050624):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1050624-2099199, default 2099199):
+700M
Value out of range.
```

Isso acontece porque o maior espaço contíguo não alocado no disco é o bloco de 512 MB que pertencia à partição 2. Sua nova partição não pode "pular por cima" da partição 3 para usar parte do espaço não alocado existente depois dela.

Mudando o tipo da partição

Ocasionalmente, pode ser necessário alterar o tipo da partição, especialmente ao lidar com discos que serão usados em outros sistemas operacionais e plataformas. Isso é feito com o comando `t`, seguido

pelo número da partição que se deseja alterar.

O tipo de partição deve ser especificado por seu código hexadecimal correspondente. Para ver uma lista de todos os códigos válidos, use o comando `l`.

Não confunda o tipo de partição com o sistema de arquivos usado nela. Embora no início houvesse uma relação entre eles, hoje não é possível presumir que isso seja verdade. Uma partição Linux, por exemplo, pode conter qualquer sistema de arquivos nativo do Linux, como *ext4* ou *ReiserFS*.

TIP As partições do Linux são do tipo `83` (Linux). As partições de troca são do tipo `82` (Linux Swap).

Gerenciando partições GUID com o GDISK

O utilitário `gdisk` é o equivalente do `fdisk` para lidar com discos particionados GPT. Na verdade, a interface foi criada a partir do `fdisk`, com um prompt interativo e os mesmos comandos (ou muito semelhantes).

Imprimindo a tabela de partição atual

O comando `p` é usado para imprimir a tabela de partição atual. A saída é mais ou menos assim:

```
Command (? for help): p
Disk /dev/sdb: 3903578 sectors, 1.9 GiB
Model: DataTraveler 2.0
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): AB41B5AA-A217-4D1E-8200-E062C54285BE
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 3903544
Partitions will be aligned on 2048-sector boundaries
Total free space is 1282071 sectors (626.0 MiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2623488         3147775   256.0 MiB   8300  Linux filesystem
```

Já de cara, notamos algumas coisas diferentes:

- Cada disco possui um Identificador de Disco (GUID) exclusivo. Este é um número hexadecimal de 128 bits, atribuído aleatoriamente quando a tabela de partição é criada. Como há 3.4×10^{38} valores possíveis para esse número, as chances de que 2 discos aleatórios tenham o mesmo GUID

são muito pequenas. O GUID pode ser usado para identificar quais sistemas de arquivos montar no momento da inicialização (e onde), eliminando a necessidade de usar o caminho do dispositivo para fazer isso (como `/dev/sdb`).

- Notou a frase `Partition table holds up to 128 entries`? É isso mesmo, dá para ter até 128 partições em um disco GPT. Por causa disso, não há necessidade de partições *primárias* e *estendidas*.
- O espaço livre é listado na última linha, então não precisamos de um equivalente ao comando `F` do fdisk.

Criando uma partição

O comando para criar uma partição é `n`, assim como em `fdisk`. A principal diferença é que, além do número da partição e do primeiro e último setores (ou o tamanho), também podemos especificar o tipo de partição durante a criação. As partições GPT suportam muitos mais tipos do que as MBR. Para ver uma lista de todos os tipos suportados, use o comando `l`.

Removendo uma partição

Para excluir uma partição, digite `d` e o número da partição. Ao contrário do `fdisk`, a primeira partição não será selecionada automaticamente se for a única no disco.

Em discos GPT, as partições podem ser facilmente reordenadas ou "classificadas" para evitar lacunas na sequência de numeração. Para isso, basta usar o comando `s`. Por exemplo, imagine um disco com a seguinte tabela de partição:

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2099200         2361343   128.0 MiB   8300  Linux filesystem
   3         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

Se excluirmos a segunda partição, a tabela fica assim:

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   3         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

Se usarmos o comando `s`, ela se torna:

```
Number  Start (sector)    End (sector)  Size       Code  Name
```

```
   1            2048         2099199   1024.0 MiB  8300  Linux filesystem
   2         2361344         2623487   128.0 MiB   8300  Linux filesystem
```

Observe que a terceira partição se tornou a segunda.

Lacuna? Que lacuna?

Ao contrário dos discos MBR, ao criar uma partição em discos GPT, o tamanho não é limitado pela quantidade máxima de espaço *contíguo* não alocado. Você pode usar até o último pedacinho de um setor livre, não importa onde ele esteja localizado no disco.

Opções de recuperação

Os discos GPT armazenam cópias de backup do cabeçalho GPT e da tabela de partição, facilitando a recuperação dos discos caso esses dados tenham sido danificados. O `gdisk` fornece recursos para auxiliar nessas tarefas de recuperação, acessadas com o comando `r`.

Para reconstruir um cabeçalho GPT principal corrompido ou uma tabela de partição, usamos `b` e `c`, respectivamente, ou usamos o cabeçalho principal e a tabela para reconstruir um backup com `d` e `e`. Também dá para converter um MBR em GPT com `f` e fazer o oposto com `g`, entre outras operações. Digite `?` No menu de recuperação para obter uma lista de todos os comandos de recuperação disponíveis e descrições sobre o que eles fazem.

Criando sistemas de arquivos

O particionamento é apenas o primeiro passo para poder usar um disco. Depois disso, é necessário formatar a partição com um sistema de arquivos antes que se possa usá-lo para armazenar dados.

Um sistema de arquivos controla como os dados são armazenados e acessados no disco. O Linux suporta muitos sistemas de arquivos, alguns nativos, como a família ext (Extended Filesystem), enquanto outros vêm de outros sistemas operacionais como o FAT do MS-DOS, o NTFS do Windows NT, HFS e HFS + do Mac OS etc.

A ferramenta padrão usada para criar um sistema de arquivos no Linux é o `mkfs`, que vem em muitos "sabores" de acordo com o sistema de arquivos com o qual ele precisa trabalhar.

Criando um sistema de arquivos ext2/ext3/ext4

O *Extended Filesystem* (ext) foi o primeiro sistema de arquivos para Linux, tendo sido substituído ao longo dos anos por novas versões chamadas ext2, ext3 e ext4. Este último é atualmente o sistema de arquivos padrão de muitas distribuições Linux.

Os utilitários `mkfs.ext2`, `mkfs.ext3` e `mkfs.ext4` são usados para criar sistemas de arquivos ext2, ext3 e ext4. De fato, todos esses "utilitários" existem apenas como links simbólicos para outro utilitário chamado `mke2fs`. O `mke2fs` altera seus padrões de acordo com o nome pelo qual é chamado. Dessa forma, todos eles têm o mesmo comportamento e parâmetros na linha de comando.

A forma de uso mais simples é:

```
# mkfs.ext2 TARGET
```

Onde `TARGET` é o nome da partição na qual o sistema de arquivos deve ser criado. Por exemplo, para criar um sistema de arquivos ext3 em `/dev/sdb1`, o comando seria:

```
# mkfs.ext3 /dev/sdb1
```

Em vez de usar o comando correspondente ao sistema de arquivos que deseja criar, você pode passar o parâmetro `-t` para `mke2fs` seguido do nome do sistema de arquivos. Por exemplo, os comandos a seguir são equivalentes e irão criar um sistema de arquivos ext4 em `/dev/sdb1`.

```
# mkfs.ext4 /dev/sdb1
# mke2fs -t ext4 /dev/sdb1
```

Parâmetros de linha de comando

O `mke2fs` suporta uma ampla gama de parâmetros e opções de linha de comando. Eis alguns dos mais significativos. Todos eles também se aplicam a `mkfs.ext2`, `mkfs.ext3` e `mkfs.ext4`:

`-b SIZE`

Define o tamanho dos blocos de dados no dispositivo para `SIZE`, que pode ser de 1024, 2048 ou 4096 bytes por bloco.

`-c`

Verifica se existem blocos defeituosos no dispositivo de destino antes de criar o sistema de arquivos. Para fazer uma verificação mais detalhada, porém muito mais lenta, aplique esse parâmetro duas vezes, como em `mkfs.ext4 -c -c TARGET`.

`-d DIRECTORY`

Copia o conteúdo do diretório especificado para a raiz do novo sistema de arquivos. Útil quando se precisa "pré-preencher" o disco com um conjunto de arquivos predefinido.

`-F`

Perigo, Will Robinson! Esta opção *força* o mke2fs a criar um sistema de arquivos, mesmo se as outras opções passadas para ele ou para o alvo forem perigosas ou não fizerem nenhum sentido. Se especificado duas vezes (como em `-F -F`), pode inclusive ser usado para criar um sistema de arquivos em um dispositivo montado ou em uso, o que é uma coisa muito, mas *muito* ruim de se fazer.

`-L VOLUME_LABEL`

Define o rótulo do volume conforme especificado em `VOLUME_LABEL`. Esse rótulo deve ter ao menos 16 caracteres.

`-n`

Esta é uma opção utilíssima que simula a criação do sistema de arquivos e mostra o que seria feito se executado sem a opção `n`. Pense nele como um modo de "teste". É bom verificar as coisas antes de realmente efetuar quaisquer alterações no disco.

`-q`

Modo silencioso. O `mke2fs` será executado normalmente, mas não produzirá nenhuma saída para o terminal. Útil ao executar `mke2fs` a partir de um script.

`-U ID`

Este parâmetro define o UUID (*Universally Unique Identifier*, ou Identificador único universal) de uma partição para o valor especificado como ID. Os UUIDs são números de 128 bits em notação hexadecimal que servem para identificar uma partição para o sistema operacional. Esse número é especificado como uma string de 32 dígitos no formato 8-4-4-4-12, ou seja, 8 dígitos, hífen, 4 dígitos, hífen, 4 dígitos, hífen, 4 dígitos, hífen, 12 dígitos, como `D249E380-7719-45A1-813C-35186883987E`. Em vez de um ID, você também pode especificar parâmetros como `clear` para remover o UUID do sistema de arquivos, `random` para usar um UUID gerado aleatoriamente, ou `time` para criar um UUID baseado em tempo.

`-V`

Modo detalhado (ou verboso), exibe muito mais informações durante a operação do que normalmente. Útil para fins de depuração.

Criando um sistema de arquivos XFS

O XFS é um sistema de arquivos de alto desempenho originalmente desenvolvido pela Silicon Graphics em 1993 para seu sistema operacional IRIX. Graças a seu desempenho e recursos de confiabilidade, ele é comumente usado para servidores e outros ambientes que exigem largura de banda alta (ou garantida) do sistema de arquivos.

As ferramentas para gerenciar os sistemas de arquivos XFS são parte do pacote `xfsprogs`. Pode ser preciso instalar esse pacote manualmente, pois ele não vem incluído por padrão em algumas distribuições Linux. Outras, como o Red Hat Enterprise Linux 7, usam o XFS como sistema de arquivos padrão.

Os sistemas de arquivos XFS são divididos em pelo menos 2 partes, uma *seção de log*, onde é mantido um log de todas as operações do sistema de arquivos (comumente chamadas de *Journal*, ou diário), e a *seção de dados*. A seção de log pode estar localizada dentro da seção de dados (que é o comportamento padrão), ou mesmo em um disco separado, para melhor desempenho e confiabilidade.

O comando mais básico para criar um sistema de arquivos XFS é `mkfs.xfs TARGET`, onde `TARGET` é a partição na qual você deseja que o sistema de arquivos seja criado. Por exemplo: `mkfs.xfs /dev/sda1`.

Como no caso do `mke2fs`, o `mkfs.xfs` suporta uma série de opções de linha de comando. Eis algumas das mais comuns.

`-b size=VALUE`

Define o tamanho do bloco no sistema de arquivos, em bytes, para aquele especificado em `VALUE`. O valor padrão é 4096 bytes (4 KiB), o mínimo é 512 e o máximo é 65536 (64 KiB).

`-m crc=VALUE`

Os parâmetros iniciados com `-m` são opções de metadados. Este aqui habilita (se `VALUE` for `1`) ou desabilita (se `VALUE` for `0`) o uso de verificações CRC32c para checar a integridade de todos os metadados no disco. Isso permite uma melhor detecção de erros e recuperação de travamentos relacionados a problemas de hardware e, portanto, ele vem habilitado por padrão. O impacto dessa verificação no desempenho costuma ser mínimo e, portanto, normalmente não há razão para desativá-lo.

`-m uuid=VALUE`

Define o UUID da partição conforme o especificado em VALUE. Lembre-se de que UUIDs são números de 32 caracteres (128 bits) em base hexadecimal, especificados em grupos de 8, 4, 4, 4 e 12 dígitos separados por hífens, como `1E83E3A3-3AE9-4AAC-BF7E-29DFFECD36C0`.

`-f`

Força a criação de um sistema de arquivos no dispositivo de destino, mesmo se um sistema de arquivos for detectado nele.

`-l logdev=DEVICE`

Coloca a seção de log do sistema de arquivos no dispositivo especificado, em vez de dentro da seção de dados.

`-l size=VALUE`

Define o tamanho da seção de log conforme o especificado em `VALUE`. O tamanho pode ser especificado em bytes, e também é possível usar sufixos como `m` ou `g`. `-l size=10m`, por exemplo, limita a seção de log a 10 Megabytes.

`-q`

Modo silencioso. Neste modo, o `mkfs.xfs` não imprime os parâmetros do sistema de arquivos que está sendo criado.

`-L LABEL`

Define o rótulo do sistema de arquivos, que pode ter no máximo 12 caracteres.

`-N`

Semelhante ao parâmetro `-n` do `mke2fs`, faz com que o `mkfs.xfs` exiba todos os parâmetros para a criação do sistema de arquivos, sem realmente criá-lo.

Criando um sistema de arquivos FAT ou VFAT

O sistema de arquivos FAT originou-se no MS-DOS e, ao longo dos anos, recebeu muitas revisões, culminando no formato FAT32 lançado em 1996 com o Windows 95 OSR2.

O VFAT é uma extensão do formato FAT16 com suporte para nomes de arquivo longos (até 255 caracteres). Ambos os sistemas de arquivos são controlados pelo mesmo utilitário, `mkfs.fat`. `mkfs.vfat` é um nome alternativo para ele.

O sistema de arquivos FAT tem desvantagens importantes que restringem seu uso em discos grandes. O FAT16, por exemplo, suporta volumes de no máximo 4 GB e um tamanho máximo de arquivo de 2 GB. O FAT32 aumenta o tamanho do volume para até 2 PB e o tamanho máximo do arquivo para 4 GB. Por causa disso, os sistemas de arquivos FAT são hoje mais comumente usados em pequenos drives USB ou cartões de memória (de até 2 GB), ou dispositivos e sistemas operacionais legados que não oferecem suporte a sistemas de arquivos mais avançados.

O comando mais básico para a criação de um sistema de arquivos FAT é `mkfs.fat TARGET`, onde `TARGET` é a partição em que você deseja que o sistema de arquivos seja criado. Por exemplo: `mkfs.fat /dev/sdc1`.

Como outros utilitários, o `mkfs.fat` suporta uma série de opções de linha de comando. Abaixo estão

as mais importantes. Uma lista completa com a descrição de cada opção pode ser lida no manual do utilitário, com o comando `man mkfs.fat`.

`-c`

Verifica se existem blocos defeituosos no dispositivo de destino antes de criar o sistema de arquivos.

`-C FILENAME BLOCK_COUNT`

Cria o arquivo especificado em `FILENAME` e em seguida cria um sistema de arquivos FAT dentro dele, produzindo assim uma "imagem de disco" vazia que pode ser posteriormente gravada em um dispositivo usando um utilitário como o `dd` ou montada como um dispositivo de loopback. Ao usar esta opção, o número de blocos no sistema de arquivos (`BLOCK_COUNT`) deve ser especificado após o nome do dispositivo.

`-F SIZE`

Seleciona o tamanho do FAT (*File Allocation Table* ou Tabela de Alocação de Arquivos), entre 12, 16 ou 32, ou seja, entre FAT12, FAT16 ou FAT32. Se isso não for especificado, o `mkfs.fat` seleciona a opção apropriada com base no tamanho do sistema de arquivos.

`-n NAME`

Define o rótulo do volume, ou nome, do sistema de arquivos. Pode ter até 11 caracteres e o padrão é sem nome.

`-v`

Modo detalhado. Imprime muito mais informações do que o normal, útil para depuração.

NOTE

O `mkfs.fat` *não pode* criar um sistema de arquivos "iniciável" De acordo com a página de manual, "isso não é tão fácil quanto você pensa" e não será implementado.

Criando um sistema de arquivos exFAT

O exFAT é um sistema de arquivos criado pela Microsoft em 2006 que aborda uma das limitações mais importantes do FAT32: o tamanho do arquivo e do disco. No exFAT, o tamanho máximo do arquivo é de 16 exabytes (no FAT32 eram 4 GB) e o tamanho máximo do disco é de 128 petabytes.

Como é bem suportado pelos três principais sistemas operacionais (Windows, Linux e macOS), trata-se de uma boa escolha nos casos em que a interoperabilidade é necessária, como em drives flash de grande capacidade, cartões de memória e discos externos. Na verdade, esse é o sistema de arquivos padrão, conforme definido pela *SD Association*, para os cartões de memória SDXC com mais de 32 GB.

O utilitário padrão para criar sistemas de arquivos exFAT é `mkfs.exfat`, que é um link para `mkexfatfs`. O comando mais básico é `mkfs.exfat TARGET`, onde `TARGET` é a partição em que você deseja que o sistema de arquivos seja criado. Por exemplo: `mkfs.exfat /dev/sdb2`.

Ao contrário dos outros utilitários discutidos nesta lição, o `mkfs.exfat` tem pouquíssimas opções de linha de comando. Elas são:

`-i VOL_ID`

Define o ID do Volume para o valor especificado em `VOL_ID`. Este é um número hexadecimal de 32 bits. Se não for definido, é criado um ID com base na hora atual.

`-n NAME`

Define o rótulo ou nome do volume. Pode ter até 15 caracteres e o padrão é sem nome.

`-p SECTOR`

Especifica o primeiro setor da primeira partição no disco. Este é um valor opcional e o padrão é zero.

`-s SECTORS`

Define o número de setores físicos por cluster de alocação. Deve ser uma potência de dois, como 1, 2, 4, 8 e assim por diante.

Conhecendo melhor o sistema de arquivos Btrfs

O Btrfs (oficialmente o *B-Tree Filesystem*: pronuncia-se "Butter FS", "Better FS" ou mesmo "Butterfuss", como preferir) é um sistema de arquivos que está em desenvolvimento desde 2007 especificamente para o Linux pela Oracle Corporation e outras empresas, incluindo Fujitsu, Red Hat, Intel e SUSE, entre outras.

Existem muitos recursos que tornam o Btrfs atraente nos sistemas modernos em que é comum haver grandes quantidades de armazenamento. Dentre esses recursos estão o suporte a múltiplos dispositivos (incluindo striping, mirroring e striping + mirroring, como em uma configuração RAID), compressão transparente, otimizações SSD, backups incrementais, instantâneos, desfragmentação online, verificações offline, suporte para subvolumes (com cotas), deduplicação e muito mais.

Por ser um sistema de arquivos *cópia em gravação* (copy-on-write), ele é muito resistente a travamentos. Além disso, o Btrfs é simples de usar e bem suportado por muitas distribuições Linux. Algumas delas, como o SUSE, o usam como sistema de arquivos padrão.

NOTE Em um sistema de arquivos tradicional, quando você deseja sobrescrever parte de um arquivo, os novos dados são colocados diretamente sobre os dados antigos que

estão substituindo. Em um sistema de arquivos *cópia em gravação* os novos dados são gravados para liberar espaço em disco, em seguida os metadados originais do arquivo são atualizados para se referir aos novos dados e somente então os dados antigos são liberados, já que não são mais necessários. Isso reduz as chances de perda de dados em caso de travamento, já que os dados antigos só são descartados depois que o sistema de arquivos tem absoluta certeza de que não são mais necessários e os novos dados estão no lugar.

Criando um sistema de arquivos Btrfs

O utilitário `mkfs.btrfs` é usado para criar um sistema de arquivos Btrfs. Se o comando for usado sem nenhuma opção, ele cria um sistema de arquivos Btrfs em um determinado dispositivo, assim:

```
# mkfs.btrfs /dev/sdb1
```

TIP Caso não tenha o utilitário `mkfs.btrfs` em seu sistema, procure por `btrfs-progs` no gerenciador de pacotes de sua distribuição.

Você pode usar `-L` para definir um rótulo (ou nome) para o seu sistema de arquivos. Os rótulos Btrfs podem ter até 256 caracteres, exceto por quebras de linha:

```
# mkfs.btrfs /dev/sdb1 -L "New Disk"
```

TIP Coloque o rótulo entre aspas (como acima) se contiver espaços.

O Btrfs tem uma coisa peculiar: é possível incluir *múltiplos* dispositivos no comando `mkfs.btrfs`. Quando passamos mais de um dispositivo, o sistema de arquivos se estenderá por todos os dispositivos, numa configuração semelhante à de um RAID ou LVM. Para especificar como os metadados serão distribuídos na matriz de disco, use o parâmetro `-m`. Os parâmetros válidos são `raid0`, `raid1`, `raid5`, `raid6`, `raid10`, `single` e `dup`.

Por exemplo, para criar um sistema de arquivos abrangendo `/dev/sdb1` e `/dev/sdc1`, concatenando as duas partições em uma maior, use:

```
# mkfs.btrfs -d single -m single /dev/sdb /dev/sdc
```

WARNING Os sistemas de arquivos abrangendo várias partições, como exemplificado acima, podem parecer vantajosos no início, mas não são uma boa ideia do ponto de vista da segurança de dados, pois uma falha em um único disco da

matriz implica em perda de dados com certeza. O risco fica maior quanto mais discos você usa, pois também haverá mais pontos de falha possíveis.

Gerenciando subvolumes

Subvolumes são como sistemas de arquivos dentro de sistemas de arquivos. Pense neles como um diretório que pode ser montado (e tratado como) um sistema de arquivos independente. Os subvolumes facilitam a organização e a administração do sistema, pois cada um deles pode ter cotas ou regras de snapshot separadas.

NOTE Subvolumes não são partições. Uma partição aloca um espaço fixo em uma unidade. Isso pode levar a problemas mais adiante, como uma partição ficando sem espaço quando outra tem bastante espaço restante. Não é assim com subvolumes, já que eles "compartilham" o espaço livre de seu sistema de arquivos raiz e aumentam conforme necessário.

Suponha que você tenha um sistema de arquivos Btrfs montado em `/mnt/disk` e deseja criar um subvolume dentro dele para armazenar seus backups. Vamos chamá-lo de `BKP`:

```
# btrfs subvolume create /mnt/disk/BKP
```

A seguir, listamos o conteúdo do sistema de arquivos `/mnt/disk`. Você verá que temos um novo diretório com o mesmo nome do subvolume.

```
$ ls -lh /mnt/disk/
total 0
drwxr-xr-x 1 root   root     0 jul 13 17:35 BKP
drwxrwxr-x 1 carol carol 988 jul 13 17:30 Images
```

NOTE Pois é, os subvolumes *também* podem ser acessados como qualquer outro diretório.

Podemos verificar se o subvolume está ativo com o comando:

```
# btrfs subvolume show /mnt/disk/BKP/
    Name:           BKP
    UUID:           e90a1afe-69fa-da4f-9764-3384f66fa32e
    Parent UUID:        -
    Received UUID:      -
    Creation time:      2019-07-13 17:35:40 -0300
    Subvolume ID:       260
```

```
    Generation:         23
    Gen at creation:    22
    Parent ID:      5
    Top level ID:       5
    Flags:          -
    Snapshot(s):
```

Você pode montar o subvolume em `/mnt/BKP` passando o parâmetro `-t btrfs -o subvol = NAME` para o comando `mount`:

```
# mount -t btrfs -o subvol=BKP /dev/sdb1 /mnt/bkp
```

NOTE O parâmetro `-t` especifica o tipo de sistema de arquivos a ser montado.

Trabalhando com instantâneos

Os instantâneos (snapshots) são como subvolumes, mas pré-preenchidos com o conteúdo do volume a partir do qual o instantâneo foi obtido.

Quando criado, um instantâneo e o volume original têm exatamente o mesmo conteúdo. Mas a partir desse momento, eles irão divergir. As alterações feitas no volume original (como arquivos adicionados, renomeados ou excluídos) não serão refletidas no instantâneo e vice-versa.

Lembre-se de que um instantâneo *não* duplica os arquivos e, inicialmente, praticamente não ocupa espaço em disco. Ele simplesmente duplica a árvore do sistema de arquivos enquanto aponta para os dados originais.

O comando para criar um snapshot é o mesmo usado para criar um subvolume, bastando adicionar o parâmetro `snapshot` após `btrfs subvolume`. O comando abaixo cria, em `/mnt/disk/snap`, um instantâneo do sistema de arquivos Btrfs montado em `/mnt/disk`:

```
# btrfs subvolume snapshot /mnt/disk /mnt/disk/snap
```

Agora, imagine que temos o seguinte conteúdo em `/mnt/disk`:

```
$ ls -lh
total 2,8M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
```

```
-rw-rw-r-- 1 carol carol 467K jul  2 11:48 LG-G8S-ThinQ-Mirror-White.jpg
-rw-rw-r-- 1 carol carol 654K jul  2 11:39 LG-G8S-ThinQ-Range.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
drwx------ 1 carol carol  366 jul 13 17:56 snap
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

Observe o diretório de snap que contém o instantâneo. Agora vamos remover alguns arquivos e verificar o conteúdo do diretório:

```
$ rm LG-G8S-ThinQ-*
$ ls -lh
total 1,7M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
drwx------ 1 carol carol  366 jul 13 17:56 snap
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

No entanto, se você verificar dentro do diretório snap, os arquivos excluídos ainda estarão lá e poderão ser restaurados, se necessário.

```
$ ls -lh snap/
total 2,8M
-rw-rw-r-- 1 carol carol 109K jul 10 16:22 Galaxy_Note_10.png
-rw-rw-r-- 1 carol carol 484K jul  5 15:01 geminoid2.jpg
-rw-rw-r-- 1 carol carol 429K jul  5 14:52 geminoid.jpg
-rw-rw-r-- 1 carol carol 467K jul  2 11:48 LG-G8S-ThinQ-Mirror-White.jpg
-rw-rw-r-- 1 carol carol 654K jul  2 11:39 LG-G8S-ThinQ-Range.jpg
-rw-rw-r-- 1 carol carol  94K jul  2 15:43 Mimoji_Comparativo.jpg
-rw-rw-r-- 1 carol carol 112K jul 10 16:20 Note10Plus.jpg
-rw-rw-r-- 1 carol carol 118K jul 11 16:36 Twitter_Down_20190711.jpg
-rw-rw-r-- 1 carol carol 324K jul  2 15:22 Xiaomi_Mimoji.png
```

Também é possível criar instantâneos somente leitura. Eles funcionam exatamente como os instantâneos graváveis, com a diferença de que o conteúdo do instantâneo não pode ser alterado, eles são "congelados" no tempo. Basta adicionar o parâmetro `-r` ao criar o instantâneo:

```
# btrfs subvolume snapshot -r /mnt/disk /mnt/disk/snap
```

Algumas palavras sobre compactação

O Btrfs suporta a compactação transparente de arquivos, com três algoritmos diferentes disponíveis para o usuário. Isso é feito automaticamente arquivo por arquivo, contanto que o sistema de arquivos seja montado com a opção `-o compress`. Os algoritmos são inteligentes o bastante para detectar arquivos incompressíveis e não tentarão compactá-los, economizando recursos do sistema. Assim, em um único diretório, você pode ter arquivos compactados e descompactados juntos. O algoritmo de compressão padrão é o ZLIB, mas o LZO (mais rápido, taxa de compressão pior) ou o ZSTD (mais rápido que o ZLIB, compressão comparável) estão disponíveis, com diversos níveis de compressão (veja o objetivo correspondente nas opções de montagem).

Gerenciando Partições com o GNU Parted

O *GNU Parted* é um editor de partição muito poderoso (daí o nome) que pode ser usado para criar, excluir, mover, redimensionar, resgatar e copiar partições. Ele trabalha com discos GPT e MBR e é capaz de cobrir quase todas as suas necessidades de gerenciamento de disco.

Existem muitos front-ends gráficos que tornam o trabalho com o `parted` muito mais fácil, como o *GParted* para ambientes de desktop baseados no GNOME e o *KDE Partition Manager* para desktops KDE. No entanto, você deve aprender a usar o `parted` na linha de comando, já que em uma configuração de servidor nunca podemos contar com a presença de um ambiente gráfico.

WARNING

Diferente de `fdisk` e `gdisk`, o parted faz alterações no disco *imediatamente* após o comando ser emitido, sem esperar por outro comando para gravar as alterações. Ao praticar, é aconselhável fazê-lo em um disco ou unidade flash vazio ou sobressalente, para que não haja risco de perda de dados caso você cometa um erro.

A maneira mais simples de começar a usar o parted é digitando `parted DEVICE`, onde `DEVICE` é o dispositivo que se deseja gerenciar (`parted /dev/sdb`). O programa inicia uma interface de linha de comando interativa, como `fdisk` e `gdisk`, com um prompt `(parted)` para você inserir os comandos.

```
# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.

(parted)
```

WARNING Tenha cuidado! Se você não especificar um dispositivo, o `parted` seleciona automaticamente o disco primário (normalmente `/dev/sda`) para operar.

Selecionando discos

Para mudar para um disco diferente do especificado na linha de comando, usamos o comando `select`, seguido pelo nome do dispositivo:

```
(parted) select /dev/sdb
Using /dev/sdb
```

Obtendo informações

O comando `print` pode ser usado para obter mais informações sobre uma partição específica ou até mesmo todos os dispositivos de bloco (discos) conectados ao seu sistema.

Para obter informações sobre a partição atualmente selecionada, basta digitar `print`:

```
(parted) print
Model: ATA CT120BX500SSD1 (scsi)
Disk /dev/sda: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End    Size    Type     File system     Flags
 1      2097kB  116GB  116GB   primary  ext4
 2      116GB   120GB  4295MB  primary  linux-swap(v1)
```

Use `print devices` para obter uma lista de todos os dispositivos de bloco conectados ao seu sistema:

```
(parted) print devices
/dev/sdb (1999MB)
/dev/sda (120GB)
/dev/sdc (320GB)
/dev/mapper/cryptswap (4294MB)
```

Para obter informações sobre todos os dispositivos conectados de uma vez, usamos `print all`. Se quiser saber quanto espaço livre existe em cada um deles, o comando é `print free`:

```
(parted) print free
Model: ATA CT120BX500SSD1 (scsi)
Disk /dev/sda: 120GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system     Flags
        32.3kB  2097kB  2065kB           Free Space
 1      2097kB  116GB   116GB   primary  ext4
        116GB   116GB   512B             Free Space
 2      116GB   120GB   4295MB  primary  linux-swap(v1)
        120GB   120GB   2098kB           Free Space
```

Criando uma tabela de partição em um disco vazio

Para criar uma tabela de partição em um disco vazio, use o comando `mklabel`, seguido pelo tipo de tabela de partição que deseja usar.

Existem muitos tipos suportados de tabelas de partição, mas os principais que você deve conhecer são `msdos`, usado aqui para se referir a uma tabela de partição MBR, e `gpt` para se referir a uma tabela de partição GPT. Para criar uma tabela de partição MBR, digite:

```
(parted) mklabel msdos
```

E para criar uma tabela de partição GPT, o comando é:

```
(parted) mklabel gpt
```

Criando uma partição

Para criar uma partição, usamos o comando `mkpart` com a sintaxe `mkpart PARTTYPE FSTYPE START END`, onde:

`PARTTYPE`

É o tipo de partição, que pode ser `primary`, `logical` ou `extended` no caso de uma tabela de partição MBR.

FSTYPE

Especifica qual sistema de arquivos será usado nesta partição. Note que `parted` *não* cria o sistema de arquivos. Ele apenas define um sinalizador na partição que informa ao sistema operacional que tipo de dados esperar dela.

START

Especifica o ponto exato no dispositivo onde a partição começa. Você pode usar unidades diferentes para especificar esse ponto. `2s` pode ser usado para se referir ao segundo setor do disco, ao passo que `1m` se refere ao início do primeiro megabyte do disco. Outras unidades comuns são `B` (bytes) e `%` (porcentagem do disco).

END

Especifica o fim da partição. Observe que este *não* é o tamanho da partição, este é *o ponto no disco onde ele termina*. Por exemplo, se você especificar `100m`, a partição terminará 100 MB após o início do disco. Podemos usar as mesmas unidades do parâmetro `START`.

Assim, o comando:

```
(parted) mkpart primary ext4 1m 100m
```

Cria uma partição primária do tipo `ext4`, começando no primeiro megabyte do disco e terminando após o 100º megabyte.

Removendo uma partição

Para remover uma partição, use o comando `rm` seguido pelo número da partição, que você pode exibir usando o comando `print`. Portanto, `rm 2` removeria a segunda partição no disco atualmente selecionado.

Recuperando partições

O `parted` é capaz de recuperar uma partição excluída. Considere que temos a seguinte estrutura de partição:

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4         primary
 3      200MB   300MB   99.6MB  ext4         primary
```

Por acidente, você removeu a partição 2 usando `rm 2`. Para recuperá-la, pode-se usar o comando `rescue`, com a sintaxe `rescue START END`, onde `START` é o local aproximado onde a partição começava e `END` o local aproximado onde terminava.

O `parted` irá analisar o disco em busca de partições e se oferecer para restaurar as que forem encontradas. No exemplo acima, a partição `2` começava em 99,6 MB e terminava em 200 MB. Portanto, você pode usar o seguinte comando para recuperar a partição:

```
(parted) rescue 90m 210m
Information: A ext4 primary partition was found at 99.6MB -> 200MB.
Do you want to add it to the partition table?

Yes/No/Cancel? y
```

A partição e seu conteúdo serão recuperados dessa forma. Note que o `rescue` só pode recuperar partições em que haja um sistema de arquivos instalado. Partições vazias não são detectadas.

Redimensionando partições ext2/3/4

O `parted` pode ser usado para redimensionar partições, tornando-as maiores ou menores. No entanto, existem algumas ressalvas:

- Durante o redimensionamento, a partição deve estar desmontada e não estar em uso.
- É preciso ter espaço livre suficiente *após* a partição para que ela possa ser ampliada no tamanho que se deseja.

O comando é `resizepart`, seguido pelo número da partição e o ponto onde deve terminar. Por exemplo, se tivermos a seguinte tabela de partição:

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   300MB   99.6MB  ext4         primary
```

Se tentarmos aumentar a partição `1` usando `resizepart`, uma mensagem de erro seria disparada, já que, com o novo tamanho, a partição `1` se sobreporia à partição `2`. No entanto, a partição `3` pode ser redimensionada, já que há espaço livre depois dela, o que pode ser verificado com o comando `print free`:

```
(parted) print free
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

 Number  Start   End     Size    File system  Name     Flags
         17.4kB  1049kB  1031kB  Free Space
 1       1049kB  99.6MB  98.6MB  ext4         primary
 2       99.6MB  200MB   100MB   ext4
 3       200MB   300MB   99.6MB  ext4         primary
         300MB   1999MB  1699MB  Free Space
```

Portanto, podemos usar o seguinte comando para redimensionar a partição 3 para 350 MB:

```
(parted) resizepart 3 350m

(parted) print
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   350MB   150MB   ext4         primary
```

Lembre-se de que o novo ponto final é especificado a partir do início do disco. Então, como a partição `3` terminava em 300 MB, agora ela precisa terminar em 350 MB.

Mas redimensionar a partição é apenas parte da tarefa. Você também precisa redimensionar o sistema de arquivos que reside nela. Para sistemas de arquivos ext2/3/4, isso é feito com o comando `resize2fs`. No caso do exemplo acima, a partição 3 ainda mostra o tamanho "antigo" quando montada:

```
$ df -h /dev/sdb3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb3        88M  1.6M   80M   2% /media/carol/part3
```

Para ajustar o tamanho, o comando `resize2fs DEVICE SIZE` pode ser usado, onde `DEVICE` corresponde à partição que você deseja redimensionar e `SIZE` é o novo tamanho. Se você omitir o parâmetro de tamanho, ele usará todo o espaço disponível da partição. Antes de redimensionar, é aconselhável desmontar a partição.

No exemplo acima:

```
$ sudo resize2fs /dev/sdb3
resize2fs 1.44.6 (5-Mar-2019)
Resizing the filesystem on /dev/sdb3 to 146212 (1k) blocks.
The filesystem on /dev/sdb3 is now 146212 (1k) blocks long.

$ df -h /dev/sdb3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb3       135M  1.6M  123M   2% /media/carol/part3
```

Para *encolher* uma partição, o processo deve ser feito na ordem inversa. *Primeiro* você redimensiona o sistema de arquivos para um tamanho novo e menor, em seguida redimensiona a própria partição usando `parted`.

WARNING Preste atenção ao reduzir partições. Se errar na ordem das coisas, você vai perder dados!

Em nosso exemplo:

```
# resize2fs /dev/sdb3 88m
resize2fs 1.44.6 (5-Mar-2019)
Resizing the filesystem on /dev/sdb3 to 90112 (1k) blocks.
The filesystem on /dev/sdb3 is now 90112 (1k) blocks long.

# parted /dev/sdb3
(parted) resizepart 3 300m
Warning: Shrinking a partition can cause data loss, are you sure
you want to continue?

Yes/No? y

(parted) print
Model: Kingston DataTraveler 2.0 (scsi)
Disk /dev/sdb: 1999MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

```
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  99.6MB  98.6MB  ext4         primary
 2      99.6MB  200MB   100MB   ext4
 3      200MB   300MB   99.7MB  ext4         primary
```

TIP Em vez de especificar um novo tamanho, você pode usar o parâmetro `-M` de `resize2fs` para ajustar o tamanho do sistema de arquivos, deixando-o grande o suficiente para os arquivos que contém.

Criando partições de troca

No Linux, o sistema pode passar páginas de memória da RAM para o disco conforme necessário, armazenando-as em um espaço separado, geralmente implementado como uma partição separada em um disco, chamada de *partição de troca* ou simplesmente *troca* (swap, em inglês). Esta partição precisa ser de um tipo específico e configurada com um utilitário apropriado (`mkswap`) antes de poder ser usada.

Para criar a partição swap usando `fdisk` ou `gdisk`, proceda como se estivesse criando uma partição normal, conforme explicado anteriormente. A única diferença é que você precisará alterar o tipo de partição para *Linux swap*.

- No `fdisk`, use o comando `t`. Selecione a partição que deseja usar e mude seu tipo para `82`. Grave as alterações no disco e saia com `w`.
- No `gdisk`, o comando para alterar o tipo de partição também é `t`, mas o código é `8200`. Grave as alterações no disco e saia com `w`.

Se estiver usando o `parted`, a partição deve ser identificada como uma partição de troca durante a criação, apenas use `linux-swap` como tipo de sistema de arquivos. Por exemplo, o comando para criar uma partição de troca de 500 MB, começando com 300 MB no disco é:

```
(parted) mkpart primary linux-swap 301m 800m
```

Assim que a partição for criada e devidamente identificada, basta usar `mkswap` seguido do dispositivo que representa a partição que deseja usar, como:

```
# mkswap /dev/sda2
```

Para habilitar a troca nesta partição, use `swapon` seguido do nome do dispositivo:

```
# swapon /dev/sda2
```

Da mesma forma, `swapoff`, seguido pelo nome do dispositivo, desabilita a troca naquele dispositivo.

O Linux também suporta o uso de *arquivos* de troca em vez de partições. Basta criar um arquivo vazio do tamanho que desejar usando `dd` e então usar `mkswap` e `swapon` tendo esse arquivo como destino.

Os comandos a seguir criam um arquivo de 1 GB chamado `myswap` no diretório atual, preenchido com zeros, e então o configuram e habilitam como um arquivo de troca.

Crie o arquivo de troca:

```
$ dd if=/dev/zero of=myswap bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 7.49254 s, 143 MB/s
```

`if =` é o arquivo de entrada, a fonte dos dados que serão gravados no arquivo. Neste caso, é o dispositivo `/dev/zero`, que fornece tantos caracteres `NULL` quanto solicitados. `of =` é o arquivo de saída, o arquivo que será criado. `bs =` é o tamanho dos blocos de dados, especificados aqui em Megabytes, e `count =` é a quantidade de blocos a serem gravados na saída. 1.024 blocos de 1 MB cada equivalem a 1 GB.

```
# mkswap myswap
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=49c53bc4-c4b1-4a8b-a613-8f42cb275b2b

# swapon myswap
```

Usando os comandos acima, este arquivo de troca será usado apenas durante a sessão atual do sistema. Se a máquina for reinicializada, o arquivo ainda estará disponível, mas não será carregado automaticamente. Você pode automatizar esse processo adicionando o novo arquivo de troca a `/etc/fstab`, que discutiremos em uma lição posterior.

TIP

Tanto o `mkswap` quanto o `swapon` vão reclamar se seu arquivo de troca tiver permissões inseguras. O sinalizador de permissão de arquivo recomendado é `0600`. O proprietário

> e o grupo devem ser `root`.

Exercícios Guiados

1. Qual esquema de particionamento deve ser usado para particionar um disco rígido de 3 TB em três partições de 1 GB? Por quê?

2. No `gdisk`, como podemos descobrir quanto espaço está disponível no disco?

3. Qual seria o comando para criar um sistema de arquivos ext3, verificando antes se há setores defeituosos, com o rótulo `MyDisk` e um UUID aleatório, no dispositivo `/dev/sdc1`?

4. Usando o `parted`, qual seria o comando para criar uma partição ext4 de 300 MB, começando com 500 MB no disco?

5. Imagine que você tenha 2 partições, uma em `/dev/sda1` e outra em `/dev/sda2`, ambas com 20 GB de tamanho. Como você pode usá-las em um único sistema de arquivos Btrfs, de forma que o conteúdo de uma partição seja automaticamente espelhado na outra, como em uma configuração RAID1? Qual será o tamanho do sistema de arquivos?

Exercícios Exploratórios

1. Considere um disco de 2 GB com uma tabela de partição MBR e o seguinte layout:

```
Disk /dev/sdb: 1.9 GiB, 1998631936 bytes, 3903578 sectors
Disk model: DataTraveler 2.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x31a83a48

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 1050623 1048576  512M 83 Linux
/dev/sdb3       2099200 3147775 1048576  512M 83 Linux
```

   É possível criar uma partição de 600 MB nele? Por quê?

2. Em um disco em `/dev/sdc`, temos uma primeira partição de 1 GB contendo cerca de 256 MB de arquivos. Usando `parted`, como podemos reduzi-la para que tenha somente espaço suficiente para os arquivos?

3. Imagine que você tem um disco em `/dev/sdb` e deseja criar uma partição swap de 1 GB no início dele. Assim, usando `parted`, você cria a partição com `mkpart primary linux-swap 0 1024M`. A seguir, você habilita o swap (troca) nesta partição com `swapon /dev/sdb1`, mas obtém a seguinte mensagem de erro:

   `swapon: /dev/sdb1: read swap header failed`

   O que deu errado?

4. Ao longo desta lição, você experimentou alguns comandos no `parted` mas, por engano, excluiu a 3ª partição do seu disco rígido. Você sabe que ela vinha depois de uma partição UEFI de 250 MB e de uma partição de troca de 4 GB, e tinha 10 GB de tamanho. Qual comando você pode usar para recuperá-la?

5. Imagine que você tenha uma partição não utilizada de 4 GB em `/dev/sda3`. Usando `fdisk`, qual seria a sequência de operações para transformá-lo em uma partição swap ativa?

Resumo

Nesta lição, você aprendeu:

- Como criar uma tabela de partição MBR em um disco com `fdisk` e como usá-la para criar e deletar partições.
- Como criar uma tabela de partição MBR em um disco com `gdisk` e como usá-la para criar e deletar partições.
- Como criar partições ext2, ext3, ext4, XFS, VFAT e exFAT.
- Como usar o `parted` para criar, excluir e recuperar partições em discos MBR e GPT.
- Como usar e redimensionar partições ext2, ext3, ext4 e Brts.
- Como criar, configurar e ativar partições de swap e arquivos de swap.

Os seguintes comandos foram abordados nesta lição:

- `fdisk`
- `gdisk`
- `mkfs.ext2`, `mkfs.ext3`, `mkfs.ext4`, `mkfs.xfs`, `mkfs.vfat` e `mkfs.exfat`
- `parted`
- `btrfs`
- `mkswap`
- `swapon` e `swapoff`

Respostas aos Exercícios Guiados

1. Qual esquema de particionamento deve ser usado para particionar um disco rígido de 3 TB em três partições de 1 GB? Por quê?

   GPT, já que o MBR suporta discos rígidos de no máximo 2 TB.

2. No `gdisk`, como podemos descobrir quanto espaço está disponível no disco?

   Usamos `p` (print). O espaço livre total será mostrado como a última linha de informação antes da própria tabela de partição.

3. Qual seria o comando para criar um sistema de arquivos ext3, verificando antes se há setores defeituosos, com o rótulo `MyDisk` e um UUID aleatório, no dispositivo `/dev/sdc1`?

   O comando seria `mkfs.ext3 -c -L MyDisk -U random /dev/sdc1`. Também seria possível usar `mke2fs -t ext3` em vez de `mkfs.ext3`.

4. Usando o `parted`, qual seria o comando para criar uma partição ext4 de 300 MB, começando com 500 MB no disco?

   O comando seria `mkpart primary ext4 500m 800m`. Lembre-se de que é necessário criar o sistema de arquivos usando `mkfs.ext4`, já que o parted não faz isso.

5. Imagine que você tenha 2 partições, uma em `/dev/sda1` e outra em `/dev/sda2`, ambas com 20 GB de tamanho. Como você pode usá-las em um único sistema de arquivos Btrfs, de forma que o conteúdo de uma partição seja automaticamente espelhado na outra, como em uma configuração RAID1? Qual será o tamanho do sistema de arquivos?

   O comando seria `mkfs.btrfs /dev/sda1 /dev/sdb1 -m raid1`. O sistema de arquivos resultante teria um tamanho de 20 GB, já que uma partição age simplesmente como um espelho da outra.

Respostas aos Exercícios Exploratórios

1. Considere um disco de 2 GB com uma tabela de partição MBR e o seguinte layout:

```
Disk /dev/sdb: 1.9 GiB, 1998631936 bytes, 3903578 sectors
Disk model: DataTraveler 2.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x31a83a48

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 1050623 1048576  512M 83 Linux
/dev/sdb3       2099200 3147775 1048576  512M 83 Linux
```

   É possível criar uma partição de 600 MB nele? Por quê?

   Não, pois não há espaço contíguo suficiente. A primeira pista de que tem algo "errado" é a lista de dispositivos: temos `/dev/sdb1` e `/dev/sdb3`, mas não `/dev/sdb2`. Então, algo está faltando.

   Em seguida, precisamos ver onde uma partição termina e onde a outra começa. A partição um termina no setor `1050623`, e a partição 2 no `2099200`. Há uma "lacuna" de 1048577 setores. Como cada setor tem 512 bytes, o total seriam 536.871.424 bytes. Dividindo por 1024, obtemos 524.288 Kilobytes. Dividimos por 1024 novamente e obtemos... 512 MB. Esse é o tamanho da "lacuna".

   Se o disco tem 2 GB, resta então um máximo de 512 MB após a partição 3. Mesmo que no total haja cerca de 1 GB não-alocado, o maior bloco contíguo tem 512 MB. Portanto, não há espaço para uma partição de 600 MB.

2. Em um disco em `/dev/sdc`, temos uma primeira partição de 1 GB contendo cerca de 256 MB de arquivos. Usando `parted`, como podemos reduzi-la para que tenha somente espaço suficiente para os arquivos?

   Essa operação teria várias etapas. Primeiro, encolhemos o sistema de arquivos usando `resize2fs`. Ao invés de especificar o novo tamanho diretamente, podemos usar o parâmetro `-M` para que ele fique "grande o bastante". Assim: `resize2fs -M /dev/sdc1`.

   Em seguida, redimensionamos a própria partição com o parted usando `resizepart`. Como se trata da primeira partição, podemos pressupor que ela começa em zero e termina em 241 MB. Assim, o comando seria `resizepart 1 241M`.

3. Imagine que você tem um disco em `/dev/sdb` e deseja criar uma partição swap de 1 GB no início dele. Assim, usando `parted`, você cria a partição com `mkpart primary linux-swap 0 1024M`. A seguir, você habilita o swap (troca) nesta partição com `swapon /dev/sdb1`, mas obtém a seguinte mensagem de erro:

   ```
   swapon: /dev/sdb1: read swap header failed
   ```

   O que deu errado?

   Você criou uma partição do tipo correto (`linux-swap`), mas lembre-se de que o `mkpart` *não cria um sistema de arquivos*. Você esqueceu de configurar a partição como espaço de troca com `mkswap` antes de usá-la.

4. Ao longo desta lição, você experimentou alguns comandos no `parted` mas, por engano, excluiu a 3ª partição do seu disco rígido. Você sabe que ela vinha depois de uma partição UEFI de 250 MB e de uma partição de troca de 4 GB, e tinha 10 GB de tamanho. Qual comando você pode usar para recuperá-la?

   Não entre em pânico, você tem todas as informações necessárias para recuperar a partição. Basta usar `rescue` e fazer as contas. Você tinha 250 MB + 4096 MB (4*1024) antes, então o ponto inicial deve ser em torno de 4346 MB. Juntando com 10.240 MB (10*1024) de tamanho, ela deve terminar em 14.586 MB. Então, `rescue 4346m 14586m` deve resolver o problema. Pode ser preciso dar um pouco de "folga" ao rescue, começando um pouco antes e terminando um pouco depois, dependendo da geometria do seu disco.

5. Imagine que você tenha uma partição não utilizada de 4 GB em `/dev/sda3`. Usando `fdisk`, qual seria a sequência de operações para transformá-lo em uma partição swap ativa?

   Primeiro, altere o tipo de partição para "Linux Swap" (82), grave suas alterações no disco e saia. Depois, use `mkswap` para configurar a partição como área de troca. Em seguida, use `swapon` para habilitá-la.

104.2 Manutenção da integridade de sistemas de arquivos

Referência ao LPI objectivo

LPIC-1 version 5.0, Exam 101, Objective 104.2

Peso

2

Áreas chave de conhecimento

- Verificar a integridade dos sistemas de arquivos.
- Monitorar os espaços livres e inodes.
- Reparar problemas simples dos sistemas de arquivos.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `du`
- `df`
- `fsck`
- `e2fsck`
- `mke2fs`
- `tune2fs`
- `xfs_repair`
- `xfs_fsr`
- `xfs_db`

104.2 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	104 Dispositivos, sistemas de arquivos do Linux, hierarquia padrão de sistemas de arquivos
Objetivo:	104.2 Manutenção da integridade dos sistemas de arquivos
Lição:	1 de 1

Introdução

Os sistemas de arquivos Linux modernos têm suporte a journaling. Isso significa que cada operação é registrada em um log interno (o *journal*, ou diário) antes de ser executada. Se a operação for interrompida devido a um erro do sistema (como kernel panic, falha de energia etc.), ela pode ser reconstituída verificando-se o diário, evitando assim a corrupção do sistema de arquivos e perda de dados.

Isso reduz muito a necessidade de verificações manuais do sistema de arquivos, mas elas ainda podem ser necessárias. Conhecer as ferramentas utilizadas para isso (e os parâmetros correspondentes) pode representar a diferença entre jantar em casa com a família ou passar a noite na sala do servidor no trabalho.

Nesta lição, discutiremos as ferramentas disponíveis para monitorar o uso do sistema de arquivos, otimizar sua operação e como verificar e reparar danos.

Verificando o uso de disco

Existem dois comandos que podem ser usados para verificar quanto espaço está sendo usado e quanto resta em um sistema de arquivos. O primeiro é `du`, que significa "disk usage" (uso do disco).

O `du` é recursivo por natureza. Em sua forma mais básica, o comando simplesmente mostra quantos blocos de 1 Kilobyte estão sendo usados pelo diretório atual e todos os seus subdiretórios:

```
$ du
4816    .
```

Isso não é muito útil, então podemos solicitar uma saída maior e "legível por humanos" adicionando o parâmetro `-h`:

```
$ du -h
4.8M    .
```

Por padrão, o `du` só mostra a contagem de uso para os diretórios (considerando todos os arquivos e subdiretórios dentro deles). Para mostrar uma contagem individual para todos os arquivos no diretório, usamos o parâmetro `-a`:

```
$ du -ah
432K    ./geminoid.jpg
508K    ./Linear_B_Hero.jpg
468K    ./LG-G8S-ThinQ-Mirror-White.jpg
656K    ./LG-G8S-ThinQ-Range.jpg
60K ./Stranger3_Titulo.png
108K    ./Baidu_Banho.jpg
324K    ./Xiaomi_Mimoji.png
284K    ./Mi_CC_9e.jpg
96K ./Mimoji_Comparativo.jpg
32K ./Xiaomi FCC.jpg
484K    ./geminoid2.jpg
108K    ./Mimoji_Abre.jpg
88K ./Mi8_Hero.jpg
832K    ./Tablet_Linear_B.jpg
332K    ./Mimoji_Comparativo.png
4.8M    .
```

O comportamento padrão é mostrar o uso de cada subdiretório e, em seguida, o uso total do diretório

atual, *incluindo* subdiretórios:

```
$ du -h
4.8M    ./Temp
6.0M    .
```

No exemplo acima, podemos ver que o subdiretório `Temp` ocupa 4,8 MB e o diretório atual, *incluindo* `Temp`, ocupa 6,0 MB. Mas quanto espaço os *arquivos* no diretório atual ocupam, excluindo os subdiretórios? Para isso temos o parâmetro `-S`:

```
$ du -Sh
4.8M    ./Temp
1.3M    .
```

TIP Lembre-se de que os parâmetros da linha de comando diferenciam maiúsculas de minúsculas: `-s` é diferente de `-S`.

Se quiser manter essa distinção entre o espaço usado pelos arquivos no diretório atual e o espaço usado pelos subdiretórios, mas *também* quiser um total geral no final, você pode adicionar o parâmetro `-c`:

```
$ du -Shc
4.8M    ./Temp
1.3M    .
6.0M    total
```

Para controlar a "profundidade" da saída de `du`, usamos o parâmetro `-d N`, onde `N` descreve os níveis. Por exemplo, se usarmos o parâmetro `-d 1`, ele mostrará o diretório atual e seus subdiretórios, mas não os subdiretórios deles.

Veja a diferença abaixo. Sem `-d`:

```
$ du -h
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

E limitando a profundidade a um nível com `-d 1`:

```
$ du -h -d1
224K    ./somedir
232K    .
```

Observe que, mesmo que `anotherdir` não esteja sendo mostrado, seu tamanho ainda está sendo levado em consideração.

Você pode querer excluir alguns tipos de arquivos da contagem, o que é feito com `--exclude="PATTERN"`, onde `PATTERN` é o padrão que deve ser correspondido. Considere este diretório:

```
$ du -ah
124K    ./ASM68K.EXE
2.0M    ./Contra.bin
36K ./fixheadr.exe
4.0K    ./README.txt
2.1M    ./Contra_NEW.bin
4.0K    ./Built.bat
8.0K    ./Contra_Main.asm
4.2M    .
```

Agora, usamos `--exclude` para filtrar todos os arquivos com a extensão `.bin`:

```
$ du -ah --exclude="*.bin"
124K    ./ASM68K.EXE
36K ./fixheadr.exe
4.0K    ./README.txt
4.0K    ./Built.bat
8.0K    ./Contra_Main.asm
180K    .
```

Observe que o total não reflete mais o tamanho dos arquivos excluídos.

Em busca de espaço livre

O `du` trabalha no nível dos arquivos. Existe outro comando que pode mostrar o uso do disco e quanto espaço está disponível no nível dos sistemas de arquivos. Esse comando é `df`.

O comando `df` fornece uma lista de todos os sistemas de arquivos disponíveis (já montados) em seu sistema, incluindo o tamanho total, quanto espaço foi usado, quanto espaço está disponível, a

porcentagem de uso e onde estão montados:

```
$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
udev             2943068         0   2943068   0% /dev
tmpfs             595892      2496    593396   1% /run
/dev/sda1      110722904  25600600  79454800  25% /
tmpfs            2979440    951208   2028232  32% /dev/shm
tmpfs               5120         0      5120   0% /run/lock
tmpfs            2979440         0   2979440   0% /sys/fs/cgroup
tmpfs             595888        24    595864   1% /run/user/119
tmpfs             595888       116    595772   1% /run/user/1000
/dev/sdb1          89111      1550     80824   2% /media/carol/part1
/dev/sdb3          83187      1550     75330   3% /media/carol/part3
/dev/sdb2          90827      1921     82045   3% /media/carol/part2
/dev/sdc1      312570036 233740356  78829680  75% /media/carol/Samsung Externo
```

No entanto, dispor das informações de tamanho em blocos de 1 KB não é lá muito amigável. Como no `du`, podemos adicionar os parâmetros `-h` para obter uma saída mais "legível por humanos":

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            2.9G     0  2.9G   0% /dev
tmpfs           582M  2.5M  580M   1% /run
/dev/sda1       106G   25G   76G  25% /
tmpfs           2.9G  930M  2.0G  32% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           2.9G     0  2.9G   0% /sys/fs/cgroup
tmpfs           582M   24K  582M   1% /run/user/119
tmpfs           582M  116K  582M   1% /run/user/1000
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1       299G  223G   76G  75% /media/carol/Samsung Externo
```

Também podemos usar o parâmetro `-i` para mostrar os inodes usados/disponíveis, em vez dos blocos:

```
$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
udev            737142    547  736595    1% /dev
```

```
tmpfs           745218    908  744310    1% /run
/dev/sda6      6766592 307153 6459439    5% /
tmpfs           745218    215  745003    1% /dev/shm
tmpfs           745218      4  745214    1% /run/lock
tmpfs           745218     18  745200    1% /sys/fs/cgroup
/dev/sda1        62464    355   62109    1% /boot
tmpfs           745218     43  745175    1% /run/user/1000
```

Um parâmetro útil é `-T`, que também imprime o tipo de cada sistema de arquivos:

```
$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
udev           devtmpfs  2.9G     0  2.9G   0% /dev
tmpfs          tmpfs     582M  2.5M  580M   1% /run
/dev/sda1      ext4      106G   25G   76G  25% /
tmpfs          tmpfs     2.9G  930M  2.0G  32% /dev/shm
tmpfs          tmpfs     5.0M     0  5.0M   0% /run/lock
tmpfs          tmpfs     2.9G     0  2.9G   0% /sys/fs/cgroup
tmpfs          tmpfs     582M   24K  582M   1% /run/user/119
tmpfs          tmpfs     582M  116K  582M   1% /run/user/1000
/dev/sdb1      ext4       88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3      ext4       82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2      ext4       89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1      fuseblk   299G  223G   76G  75% /media/carol/Samsung Externo
```

Conhecendo o tipo do sistema de arquivos, podemos filtrar a saída. A ideia é mostrar apenas sistemas de arquivos de um determinado tipo com `-t TYPE` ou excluir sistemas de arquivos de um determinado tipo com `-x TYPE`, como nos exemplos abaixo.

Excluindo os sistemas de arquivos `tmpfs`:

```
$ df -hx tmpfs
Filesystem      Size  Used Avail Use% Mounted on
udev            2.9G     0  2.9G   0% /dev
/dev/sda1       106G   25G   76G  25% /
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
/dev/sdc1       299G  223G   76G  75% /media/carol/Samsung Externo
```

Exibindo apenas sistemas de arquivos `ext4`:

```
$ df -ht ext4
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       106G   25G   76G  25% /
/dev/sdb1        88M  1.6M   79M   2% /media/carol/part1
/dev/sdb3        82M  1.6M   74M   3% /media/carol/part3
/dev/sdb2        89M  1.9M   81M   3% /media/carol/part2
```

Também podemos personalizar a saída de `df`, selecionando o que deve ser exibido e em que ordem, usando o parâmetro `--output=` seguido por uma lista separada por vírgulas dos campos que desejamos exibir. Alguns dos campos disponíveis são:

`source`
: O dispositivo correspondente ao sistema de arquivos.

`fstype`
: O tipo de sistema de arquivos.

`size`
: O tamanho total do sistema de arquivos.

`used`
: Quanto espaço está sendo usado.

`avail`
: Quanto espaço está disponível.

`pcent`
: A porcentagem de uso.

`target`
: Onde o sistema de arquivos é montado (ponto de montagem).

Se quiser uma saída mostrando o destino, a fonte, o tipo e o uso, você pode usar:

```
$ df -h --output=target,source,fstype,pcent
Mounted on                   Filesystem     Type     Use%
/dev                         udev           devtmpfs   0%
/run                         tmpfs          tmpfs      1%
/                            /dev/sda1      ext4      25%
/dev/shm                     tmpfs          tmpfs     32%
```

```
/run/lock                    tmpfs          tmpfs      0%
/sys/fs/cgroup               tmpfs          tmpfs      0%
/run/user/119                tmpfs          tmpfs      1%
/run/user/1000               tmpfs          tmpfs      1%
/media/carol/part1           /dev/sdb1      ext4       2%
/media/carol/part3           /dev/sdb3      ext4       3%
/media/carol/part2           /dev/sdb2      ext4       3%
/media/carol/Samsung Externo  /dev/sdc1     fuseblk   75%
```

O `df` também pode ser usado para verificar as informações do inode, passando os seguintes campos para `--output=`:

`itotal`

O número total de inodes no sistema de arquivos.

`iused`

O número de inodes usados no sistema de arquivos.

`iavail`

O número de inodes disponíveis no sistema de arquivos.

`ipcent`

A porcentagem de inodes usados no sistema de arquivos.

Por exemplo:

```
$ df --output=source,fstype,itotal,iused,ipcent
Filesystem     Type      Inodes  IUsed IUse%
udev           devtmpfs  735764    593    1%
tmpfs          tmpfs     744858   1048    1%
/dev/sda1      ext4     7069696 318651    5%
tmpfs          tmpfs     744858    222    1%
tmpfs          tmpfs     744858      3    1%
tmpfs          tmpfs     744858     18    1%
tmpfs          tmpfs     744858     22    1%
tmpfs          tmpfs     744858     40    1%
```

Manutenção de sistemas de arquivos ext2, ext3 e ext4

Para procurar por erros em um sistema de arquivos (e, com sorte, corrigi-los), o Linux oferece o utilitário `fsck` (pense em "filesystem check", verificação do sistema de arquivos, e você nunca mais

esquecerá esse nome). Em sua forma mais básica, ele é invocado com `fsck` seguido da localização do sistema de arquivos que se deseja verificar:

```
# fsck /dev/sdb1
fsck from util-linux 2.33.1
e2fsck 1.44.6 (5-Mar-2019)
DT_2GB: clean, 20/121920 files, 369880/487680 blocks
```

WARNING NUNCA execute `fsck` (ou utilitários relacionados) em um sistema de arquivos montado. Se isso for feito, pode haver perda de dados.

O `fsck` em si não verifica o sistema de arquivos, mas apenas chama para isso o utilitário apropriado para o tipo de sistema de arquivos em questão. No exemplo acima, como um tipo de sistema de arquivos não foi especificado, o `fsck` pressupôs se tratar de um sistema de arquivos ext2/3/4 por padrão, e chamou `e2fsck`.

Para especificar um sistema de arquivos, use a opção `-t`, seguida pelo nome do sistema de arquivos, como em `fsck -t vfat /dev/sdc`. Alternativamente, podemos chamar diretamente um utilitário específico ao sistema de arquivos, como o `fsck.msdos` para sistemas de arquivos FAT.

TIP Digite `fsck` seguido por `Tab` duas vezes para ver uma lista de todos os comandos relacionados em seu sistema.

O `fsck` aceita alguns argumentos de linha de comando. Estes são alguns dos mais comuns:

`-A`

Verifica todos os sistemas de arquivos listados em `/etc/fstab`.

`-C`

Exibe uma barra de progresso ao verificar um sistema de arquivos. Atualmente funciona apenas em sistemas de arquivos ext2/3/4.

`-N`

Imprime na tela o que seria feito e sai, sem de fato verificar o sistema de arquivos.

`-R`

Quando usado em conjunto com `-A`, ele pula a verificação do sistema de arquivos raiz.

`-V`

Modo detalhado, imprime mais informações do que o normal durante a operação. Útil para

depuração.

O utilitário específico para sistemas de arquivos ext2, ext3 e ext4 é o `e2fsck`, também chamado `fsck.ext2`, `fsck.ext3` e `fsck.ext4` (esses três são apenas links para `e2fsck`). Por padrão, ele é executado no modo interativo: quando um erro é encontrado no sistema de arquivos, ele para e pergunta ao usuário o que fazer. O usuário deve digitar `y` para corrigir o problema, `n` para deixá-lo sem solução ou `a` para corrigir o problema atual e todos os subsequentes.

É claro que sentar em frente a um terminal esperando o `e2fsck` perguntar o que fazer não é um uso produtivo do seu tempo, especialmente se você estiver lidando com um grande sistema de arquivos. Dessa forma, existem opções que fazem com que o `e2fsck` seja executado em modo não interativo:

`-p`

Essa opção tenta corrigir automaticamente quaisquer erros encontrados. Se for encontrado um erro que requeira intervenção do administrador do sistema, o `e2fsck` fornecerá uma descrição do problema e sairá.

`-y`

Responde `y` (sim) a todas as questões.

`-n`

O oposto de `-y`. Além de responder `n` (não) a todas as questões, faz com que o sistema de arquivos seja montado somente para leitura e, portanto, não possa ser modificado.

`-f`

Força o `e2fsck` a verificar um sistema de arquivos mesmo se ele estiver marcado como "limpo", ou seja, que foi corretamente desmontado.

Ajustando um sistema de arquivos ext

Os sistemas de arquivos ext2, ext3 e ext4 têm diversos parâmetros que podem ser ajustados ou "refinados" pelo administrador do sistema para melhor atender às necessidades do sistema. O utilitário usado para exibir ou modificar esses parâmetros se chama `tune2fs`.

Para ver os parâmetros atuais de qualquer sistema de arquivos, use o parâmetro `-l` seguido pelo dispositivo que representa a partição. O exemplo abaixo mostra a saída desse comando na primeira partição do primeiro disco (`/dev/sda1`) de uma máquina:

```
# tune2fs -l /dev/sda1
tune2fs 1.44.6 (5-Mar-2019)
Filesystem volume name:   <none>
```

```
Last mounted on:          /
Filesystem UUID:          6e2c12e3-472d-4bac-a257-c49ac07f3761
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
needs_recovery extent 64bit flex_bg sparse_super large_file huge_file dir_nlink
extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              7069696
Block count:              28255605
Reserved block count:     1412780
Free blocks:              23007462
Free inodes:              6801648
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      1024
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Mon Jun 17 13:49:59 2019
Last mount time:          Fri Jun 28 21:14:38 2019
Last write time:          Mon Jun 17 13:53:39 2019
Mount count:              8
Maximum mount count:      -1
Last checked:             Mon Jun 17 13:49:59 2019
Check interval:           0 (<none>)
Lifetime writes:          20 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:           256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
First orphan inode:       5117383
Default directory hash:   half_md4
Directory Hash Seed:      fa95a22a-a119-4667-a73e-78f77af6172f
```

```
Journal backup:           inode blocks
Checksum type:            crc32c
Checksum:                 0xe084fe23
```

Os sistemas de arquivos ext têm *contagens de montagem*. A contagem é aumentada em 1 a cada vez que o sistema de arquivos é montado, e quando um valor limite (a *contagem máxima de montagem*) é alcançado, o sistema será verificado automaticamente com `e2fsck` na próxima inicialização.

A contagem máxima de montagens pode ser definida com o parâmetro `-c N`, onde `N` é o número de vezes que o sistema de arquivos pode ser montado sem ser verificado. O parâmetro `-C N` define o número de vezes que o sistema foi montado com o valor de `N`. Observe que os parâmetros da linha de comando diferenciam maiúsculas de minúsculas, então `-c` é diferente de `-C`.

Também é possível definir um intervalo de tempo entre as verificações com o parâmetro `-i`, seguido por um número e as letras `d` para dias, `m` para meses e `y` para anos. Por exemplo, `-i 10d` verificaria o sistema de arquivos na reinicialização seguinte a cada 10 dias. Use zero como valor para desabilitar este recurso.

`-L` pode ser usado para definir um rótulo para o sistema de arquivos. Esse rótulo pode ter até 16 caracteres. O parâmetro `-U` define o UUID do sistema de arquivos, que é um número hexadecimal de 128 bits. No exemplo acima, o UUID é `6e2c12e3-472d-4bac-a257-c49ac07f3761`. Tanto o rótulo quanto o UUID podem ser usados no lugar do nome do dispositivo (como `/dev/sda1`) para montar o sistema de arquivos.

A opção `-e BEHAVIOUR` define o comportamento do kernel quando um erro é encontrado no sistema de arquivos. Existem três comportamentos possíveis:

`continue`
: Continua a execução normalmente.

`remount-ro`
: Remonta o sistema de arquivos como somente leitura.

`panic`
: Causa um kernel panic.

O comportamento padrão é `continue`. `remount-ro` pode ser útil em aplicativos com dados sensíveis, pois irá interromper imediatamente as gravações no disco, evitando mais erros potenciais.

Os sistemas de arquivos ext3 são basicamente sistemas de arquivos ext2 com um diário. Usando o `tune2fs`, podemos adicionar um diário a um sistema de arquivos ext2, convertendo-o assim em ext3.

O procedimento é simples: basta passar o parâmetro `-j` para `tune2fs`, seguido do dispositivo que contém o sistema de arquivos:

```
# tune2fs -j /dev/sda1
```

Posteriormente, ao montar o sistema de arquivos convertido, não se esqueça de definir o tipo para `ext3` para que o diário possa ser usado.

Ao lidar com sistemas de arquivos com journaling, o parâmetro `-J` permite usar parâmetros extras para definir algumas opções de diário, como `-J size=` para definir o tamanho do diário (em megabytes), `-J location=` para especificar onde o diário deve ser armazenado (seja um bloco específico ou uma posição específica no disco com sufixos como `M` ou `G`) e até mesmo colocar o diário em um dispositivo externo com `-J device=`.

Para especificar diversos parâmetros ao mesmo tempo, eles devem ser separados por vírgula. Por exemplo: `-J size=10,location=100M,device=/dev/sdb1` criam um diário (Journal) de 10 MB na posição 100 MB do dispositivo `/dev/sdb1`.

WARNING

O `tune2fs` tem uma opção de "força bruta", `-f`, que o força a completar uma operação mesmo que sejam encontrados erros. Nem é preciso dizer que esse recurso deve ser usado com extrema cautela.

Manutenção de sistema de arquivos XFS

Para os sistemas de arquivos XFS, o equivalente a `fsck` é `xfs_repair`. Se você suspeitar que algo está errado com o sistema de arquivos, a primeira coisa a fazer é verificar se ocorreram danos.

Isso pode ser feito passando o parâmetro `-n` para `xfs_repair`, seguido pelo dispositivo que contém o sistema de arquivos. O parâmetro `-n` significa "no modify" : o sistema de arquivos será verificado e os erros relatados, mas nenhum reparo será feito:

```
# xfs_repair -n /dev/sdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but do not clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
```

```
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 1
        - agno = 3
        - agno = 0
        - agno = 2
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

Se forem encontrados erros, você pode prosseguir com os reparos sem o parâmetro `-n`, da seguinte forma: `xfs_repair /dev/sdb1`.

`xfs_repair` aceita uma série de opções de linha de comando. Dentre elas:

`-l LOGDEV` and `-r RTDEV`

Necessários se o sistema de arquivos tem log externo e seções em tempo real. Neste caso, substitua `LOGDEV` e `RTDEV` pelos dispositivos correspondentes.

`-m N`

Usado para limitar o uso de memória de `xfs_repair` para `N` megabytes, algo que pode ser útil nas configurações do servidor. De acordo com a página do manual, por padrão `xfs_repair` adapta seu uso de memória conforme necessário, até 75% da RAM física do sistema.

`-d`

O modo "dangerous" (perigoso) permite reparar sistemas de arquivos montados como apenas leitura,

`-v`

Você deve ter adivinhado: modo verboso. Cada vez que este parâmetro é usado, a "verbosidade" é aumentada (por exemplo,`-v -v` imprime mais informações do que apenas `-v`).

Observe que `xfs_repair` não é capaz de reparar sistemas de arquivos com um log "sujo". É possível

"zerar" um log corrompido com o parâmetro `-L`, mas tenha em mente que este é um *último recurso*, pois pode resultar em corrupção do sistema de arquivos e perda de dados.

Para depurar um sistema de arquivos XFS, o utilitário `xfs_db` pode ser usado, como em `xfs_db /dev/sdb1`. Ele serve principalmente para inspecionar diversos elementos e parâmetros do sistema de arquivos.

Este utilitário tem um prompt interativo, como o `parted`, com muitos comandos internos. Um sistema de ajuda também está disponível: digite `help` para ver uma lista de todos os comandos, e `help` seguido do nome do comando para ver mais informações sobre o comando.

Outro utilitário útil é o `xfs_fsr`, que pode ser usado para reorganizar ("desfragmentar") um sistema de arquivos XFS. Quando executado sem nenhum argumento extra, ele roda por duas horas e tenta desfragmentar todos os sistemas de arquivos XFS graváveis e montados listados no arquivo `/etc/mtab/`. Pode ser necessário instalar esse utilitário usando o gerenciador de pacotes de sua distribuição Linux, pois ele nem sempre faz parte de uma instalação padrão. Para obter mais informações, consulte a página do manual correspondente.

Exercícios Guiados

1. Usando `du`, como podemos verificar quanto espaço está sendo usado apenas pelos arquivos no diretório atual?

2. Usando `df`, liste as informações de cada sistema de arquivos ext4, incluindo na saída os seguintes campos, nesta ordem: dispositivo, ponto de montagem, número total de inodes, número de inodes disponíveis, porcentagem de espaço livre.

3. Qual é o comando para executar o `e2fsck` em `/dev/sdc1` no modo não interativo, tentando corrigir automaticamente a maioria dos erros?

4. Suponha que `/dev/sdb1` seja um sistema de arquivos ext2. Como podemos convertê-lo para ext3 e, ao mesmo tempo, redefinir sua contagem de montagens e alterar seu rótulo para `UserData`?

5. Como verificar se há erros em um sistema de arquivos XFS, *sem* reparar qualquer dano encontrado?

Exercícios Exploratórios

1. Considere um sistema de arquivos ext4 em `/dev/sda1` com os parâmetros a seguir, obtidos com `tune2fs`:

```
Mount count:              8
Maximum mount count:      -1
```

   O que acontecerá na próxima inicialização se o comando `tune2fs -c 9 /dev/sda1` for emitido?

2. Considere a saída a seguir de `du -h`:

```
$ du -h
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

   Quanto espaço está ocupado apenas pelos arquivos no diretório atual? Como poderíamos reescrever o comando para mostrar essas informações com mais clareza?

3. O que aconteceria ao sistema de arquivos ext2 `/dev/sdb1` se o comando abaixo fosse emitido?

```
# tune2fs -j /dev/sdb1 -J device=/dev/sdc1 -i 30d
```

4. Como podemos verificar se há erros em um sistema de arquivos XFS em `/dev/sda1` que tem uma seção de log em `/dev/sdc1`, mas sem fazer nenhum reparo?

5. Qual a diferença entre os parâmetros `-T` e `-t` do `df`?

Resumo

Nesta lição, você aprendeu:

- Como verificar o espaço usado e livre em um sistema de arquivos.
- Como ajustar a saída de `df` para atender às suas necessidades.
- Como verificar a integridade e reparar um sistema de arquivos com `fsck` e `e2fsck`.
- Como ajustar um sistema de arquivos ext com `tune2fs`.
- Como verificar e reparar sistemas de arquivos XFS com `xfs_repair`.

Os seguintes comandos foram abordados nesta lição:

`du`
: Exibe a quantidade de espaço em disco em uso em um sistema de arquivos.

`df`
: Exibe a quantidade de espaço em disco disponível (livre) em um sistema de arquivos.

`fsck`
: Utilitário para verificação de erros do sistema de arquivos.

`e2fsck`
: Utilitário para verificação de erros específico aos sistemas de arquivos estendidos (ext2/3/4).

`tune2fs`
: Modifica parâmetros do sistema de arquivos em um sistema de arquivos estendido (ext2/3/4).

`xfs_repair`
: Equivalente a `fsck` para sistemas de arquivos XFS.

`xfs_db`
: Utilitário usado para visualizar diversos parâmetros de um sistema de arquivos XFS.

Respostas aos Exercícios Guiados

1. Usando `du`, como podemos verificar quanto espaço está sendo usado apenas pelos arquivos no diretório atual?

   Primeiro, use o parâmetro `-S` para separar a saída do diretório atual de seus subdiretórios. Em seguida, use `-d 0` para limitar a profundidade da saída a zero, indicando "sem subdiretórios". Não se esqueça do `-h` para obter uma saída em formato "legível para humanos":

   ```
   $ du -S -h -d 0
   ```

   or

   ```
   $ du -Shd 0
   ```

2. Usando `df`, liste as informações de cada sistema de arquivos ext4, incluindo na saída os seguintes campos, nesta ordem: dispositivo, ponto de montagem, número total de inodes, número de inodes disponíveis, porcentagem de espaço livre.

   Podemos filtrar sistemas de arquivos com a opção `-t` seguida pelo nome do sistema de arquivos. Para obter a saída necessária, use `--output=source,target,itotal,iavail,pcent`. Assim, a resposta é:

   ```
   $ df -t ext4 --output=source,target,itotal,iavail,pcent
   ```

3. Qual é o comando para executar o `e2fsck` em `/dev/sdc1` no modo não interativo, tentando corrigir automaticamente a maioria dos erros?

   O parâmetro para tentar corrigir automativamente a maioria dos erros é `-p`. Então, a resposta é:

   ```
   # e2fsck -p /dev/sdc1
   ```

4. Suponha que `/dev/sdb1` seja um sistema de arquivos ext2. Como podemos convertê-lo para ext3 e, ao mesmo tempo, redefinir sua contagem de montagens e alterar seu rótulo para `UserData`?

   Lembre-se de que para converter um sistema de arquivos ext2 em ext3 basta adicionar um diário, o que pode ser feito com o parâmetro `-j`. Para redefinir a contagem de montagem, use `-c 0`. Para alterar o rótulo use `-L UserData`. A resposta correta é:

```
# tune2fs -j -c 0 -L UserData /dev/sdb1
```

5. Como verificar se há erros em um sistema de arquivos XFS, *sem* reparar qualquer dano encontrado?

   Use o parâmetro `-n` parameter, como em `xfs -n`, seguido pelo dispositivo correspondente.

Respostas aos Exercícios Exploratórios

1. Considere um sistema de arquivos ext4 em `/dev/sda1` com os parâmetros a seguir, obtidos com `tune2fs`:

```
Mount count:              8
Maximum mount count:      -1
```

   O que acontecerá na próxima inicialização se o comando `tune2fs -c 9 /dev/sda1` for emitido?

   O comando define a contagem de montagem máxima do sistema de arquivos para 9. Como a contagem atual de montagem é 8, a próxima inicialização do sistema causará uma verificação do sistema de arquivos.

2. Considere a saída a seguir de `du -h`:

```
$ du -h
216K    ./somedir/anotherdir
224K    ./somedir
232K    .
```

   Quanto espaço está ocupado apenas pelos arquivos no diretório atual? Como poderíamos reescrever o comando para mostrar essas informações com mais clareza?

   Do total de 232 K usados, 224 K são ocupados pelo diretório `somedir` e seus subdiretórios. Assim, se excluirmos esses, restam 8K sendo ocupados pelos arquivos no diretório atual. Essa informação pode ser exibida mais claramente com o parâmetro `-S`, que separa os diretórios na contagem.

3. O que aconteceria ao sistema de arquivos ext2 `/dev/sdb1` se o comando abaixo fosse emitido?

```
# tune2fs -j /dev/sdb1 -J device=/dev/sdc1 -i 30d
```

   Um diário seria adicionado a `/dev/sdb1`, convertendo-o em ext3. O diário será armazenado no dispositivo `/dev/sdc1` e o sistema de arquivos será verificado a cada 30 dias.

4. Como podemos verificar se há erros em um sistema de arquivos XFS em `/dev/sda1` que tem uma seção de log em `/dev/sdc1`, mas sem fazer nenhum reparo?

Use `xfs_repair`, seguido por `-l /dev/sdc1` para indicar o dispositivo que contém a seção de log e `-n` para evitar que sejam feitas alterações.

```
# xfs_repair -l /dev/sdc1 -n
```

5. Qual a diferença entre os parâmetros `-T` e `-t` do `df`?

   O parâmetro `-T` inclui o tipo de cada sistema de arquivos na saída de `df`. `-t` é um filtro e mostra apenas sistemas de arquivos do tipo solicitado na saída, excluindo todos os outros.

104.3 Controle da montagem e desmontagem dos sistemas de arquivos

Referência ao LPI objectivo

LPIC-1 version 5.0, Exam 101, Objective 104.3

Peso

3

Áreas chave de conhecimento

- Montar e desmontar manualmente sistemas de arquivos.
- Configurar a montagem dos sistemas de arquivos no início do sistema.
- Configurar sistemas de arquivos removíveis e montáveis pelo usuário.
- Utilização de etiquetas (labels) e UUIDs para identificar e montar sistemas de arquivos.
- Noções de unidades de montagem do systemd.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `/etc/fstab`
- `/media/`
- `mount`
- `umount`
- `blkid`
- `lsblk`

104.3 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	104 Dispositivos, sistemas de arquivos do Linux, hierarquia padrão de sistemas de arquivos
Objetivo:	104.3 Controlar a montagem e desmontagem dos sistema de arquivos
Lição:	1 de 1

Introdução

Até aqui, você aprendeu a particionar discos e a criar e manter sistemas de arquivos neles. No entanto, antes que um sistema de arquivos possa ser acessado no Linux, ele precisa ser *montado*.

Montar significa anexar o sistema de arquivos em um ponto específico da árvore de diretórios do sistema, chamado *ponto de montagem*. Os sistemas de arquivos podem ser montados manual ou automaticamente e há muitas maneiras de fazê-lo. Veremos algumas delas nesta lição.

Montando e desmontando sistemas de arquivos

O comando para montar manualmente um sistema de arquivos é `mount` e sua sintaxe é:

```
mount -t TYPE DEVICE MOUNTPOINT
```

Onde:

`TYPE`

O tipo de sistema de arquivos sendo montado (p. ex. ext4, btrfs, exfat, etc.).

`DEVICE`

O nome da partição que contém o sistema de arquivos (p. ex. `/dev/sdb1`)

`MOUNTPOINT`

Onde o sistema de arquivos será montado. O diretório de montagem não precisa estar vazio, embora precise existir. Porém, quaisquer arquivos que ele contiver estarão inacessíveis por nome enquanto o sistema de arquivos estiver montado.

Por exemplo, para montar uma unidade flash USB contendo um sistema de arquivos exFAT localizado em `/dev/sdb1` em um diretório chamado `flash` em seu diretório inicial, você usaria:

```
# mount -t exfat /dev/sdb1 ~/flash/
```

TIP

Muitos sistemas Linux usam o shell Bash, e nesse caso o til `~` no caminho para o ponto de montagem é uma abreviação para o diretório inicial do usuário atual. Se o nome do usuário atual for `john`, por exemplo, ele será substituído por `/home/john`.

Após a montagem, o conteúdo do sistema de arquivos estará acessível no diretório `~/flash`:

```
$  ls -lh ~/flash/
total 469M
-rwxrwxrwx 1 root root 454M jul 19 09:49 lineage-16.0-20190711-MOD-quark.zip
-rwxrwxrwx 1 root root  16M jul 19 09:44 twrp-3.2.3-mod_4-quark.img
```

Listando sistemas de arquivos montados

Se você digitar apenas `mount`, obterá uma lista de todos os sistemas de arquivos atualmente montados em seu sistema. Essa lista pode ser bastante extensa, já que, além dos discos anexados ao sistema, ela também conterá diversos sistemas de arquivos de tempo de execução na memória que servem a vários propósitos. Se quiser filtrar a saída, use o parâmetro `-t` para listar apenas os sistemas de arquivos do tipo correspondente, como abaixo:

```
# mount -t ext4
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
```

Podemos especificar vários sistemas de arquivos de uma vez, separando-os com uma vírgula:

```
# mount -t ext4,fuseblk
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
/dev/sdb1 on /home/carol/flash type fuseblk
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other,blks
ize=4096) [DT_8GB]
```

A saída nos exemplos acima pode ser descrita no formato:

```
SOURCE on TARGET type TYPE OPTIONS
```

Onde `SOURCE` é a partição que contém o sistema de arquivos, `TARGET` é o diretório onde ele está montado, `TYPE` é o tipo do sistema de arquivos e `OPTIONS` são as opções passadas para o comando `mount` no momento da montagem.

Parâmetros adicionais da linha de comando

Diversos parâmetros de linha de comando podem ser usados com `mount`. Alguns dos mais frequentes são:

`-a`

Monta todos os sistemas de arquivos listados no arquivo `/etc/fstab` (trataremos disso na próxima seção).

`-o` ou `--options`

Passa uma lista de *opções de montagem* separadas por vírgula para o comando de montagem, que pode mudar a forma como o sistema de arquivos será montado. Falaremos disso junto com `/etc/fstab`.

`-r` ou `-ro`

Monta o sistema de arquivos como somente leitura.

`-w` ou `-rw`

Monta o sistema de arquivos como gravável.

Para desmontar um sistema de arquivos, use o comando `umount`, seguido pelo nome do dispositivo ou ponto de montagem. Considerando o exemplo acima, os comandos abaixo são intercambiáveis:

```
# umount /dev/sdb1
# umount ~/flash
```

Alguns dos parâmetros da linha de comando para `umount` são:

`-a`

Desmonta todos os sistemas de arquivos listados em `/etc/fstab`.

`-f`

Força a desmontagem de um sistema de arquivos. Pode ser útil se você tiver montado um sistema de arquivos remoto que se tornou inacessível.

`-r`

Se o sistema de arquivos não puder ser desmontado, esse comando tenta torná-lo somente leitura.

Como lidar com arquivos abertos

Ao desmontar um sistema de arquivos, pode aparecer a mensagem de erro `target is busy`. Isso acontece quando algum arquivo do sistema de arquivos está aberto. No entanto, nem sempre a localização de um arquivo aberto é óbvia, nem o que está acessando o sistema de arquivos.

Nesses casos, podemos usar o comando `lsof` seguido do nome do dispositivo que contém o sistema de arquivos para ver uma lista de processos que o acessam e quais arquivos estão abertos. Por exemplo:

```
# umount /dev/sdb1
umount: /media/carol/External_Drive: target is busy.

# lsof /dev/sdb1
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
evince  3135 carol   16r   REG   8,17 21881768 5195
/media/carol/External_Drive/Documents/E-Books/MagPi40.pdf
```

`COMMAND` é o nome do executável que abriu o arquivo e `PID` é o número do processo. `NAME` é o nome do arquivo que está aberto. No exemplo acima, o arquivo `MagPi40.pdf` foi aberto pelo programa `evince` (um visualizador de PDF). Se fecharmos o programa, poderemos desmontar o sistema de arquivos.

NOTE

Antes que a saída de `lsof` apareça, os usuários de `GNOME` poderão ver uma mensagem de aviso na janela do terminal.

```
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system
/run/user/1000/gvfs
Output information may be incomplete.
```

O `lsof` tenta processar todos os sistemas de arquivos montados. Essa mensagem de aviso surge porque o `lsof` encontrou um sistema de arquivos virtual `GNOME` (GVFS). Esse é um caso especial de sistema de arquivos no espaço de usuário (`FUSE`). Ele age como uma ponte entre `GNOME`, suas APIs e o kernel. Ninguém—nem mesmo o root—pode acessar um desses sistemas de arquivo, exceto pelo proprietário que o montou (neste caso, `GNOME`). Você pode ignorar esse aviso.

Onde montar?

Você pode montar um sistema de arquivos em qualquer lugar que desejar. No entanto, existem algumas práticas recomendadas para facilitar a administração do sistema.

Tradicionalmente, `/mnt` era o diretório sob o qual todos os dispositivos externos eram montados e, dentro dele, existia uma série de "pontos de ancoragem" pré-configurados para dispositivos comuns, como drives de CD-ROM (`/mnt/cdrom`) e disquetes (`/mnt/floppy`).

Ele foi substituído por `/media`, que agora é o ponto de montagem padrão para qualquer mídia removível pelo usuário (por exemplo, discos externos, pendrives, leitores de cartão de memória, etc.) conectada ao sistema.

Na maioria das distribuições Linux e ambientes de desktop modernos, os dispositivos removíveis são montados automaticamente em `/media/USER/LABEL` quando conectados ao sistema, sendo `USER` o nome de usuário e `LABEL` o nome do dispositivo. Por exemplo, um drive flash USB com o nome `FlashDrive` conectado pelo usuário `john` seria montado em `/media/john/FlashDrive/`. A maneira como isso é feito varia de acordo com o ambiente de desktop. Dito isto, sempre que você precisar montar *manualmente* um sistema de arquivos, é melhor fazer isso em `/mnt`.

Montagem de sistemas de arquivos na inicialização

O arquivo `/etc/fstab` contém descrições sobre os sistemas de arquivos que podem ser montados. Trata-se de um arquivo de texto em que cada linha descreve um sistema de arquivos a ser montado, com seis campos por linha, na seguinte ordem:

```
FILESYSTEM MOUNTPOINT TYPE OPTIONS DUMP PASS
```

Onde:

`FILESYSTEM`

O dispositivo que contém o sistema de arquivos a ser montado. Em vez do dispositivo, você pode especificar o UUID ou rótulo da partição, algo que discutiremos mais tarde.

`MOUNTPOINT`

Onde o sistema de arquivos será montado.

`TYPE`

O tipo de sistema de arquivos.

`OPTIONS`

Opções de montagem que serão passadas para `mount`.

`DUMP`

Indica se qualquer sistema de arquivos ext2, ext3 ou ext4 deve ser considerado para backup pelo comando `dump`. Normalmente o valor é zero, o que significa que devem ser ignorados.

`PASS`

Quando diferente de zero, define a ordem na qual os sistemas de arquivos serão checados na inicialização. Normalmente é zero.

Por exemplo, a primeira partição do primeiro disco de uma máquina pode ser descrita como:

```
/dev/sda1  /  ext4  noatime,errors
```

As opções de montagem em `OPTIONS` são uma lista de parâmetros separados por vírgulas, que podem ser genéricos ou específicos ao sistema de arquivos. Entre os genéricos temos:

`atime` e `noatime`

Por padrão, cada vez que um arquivo é lido, a informação de data e hora de acesso é atualizada. Se essa opção for desativada (com `noatime`), a E/S do disco fica mais veloz. Não confundir com a hora de modificação, que é atualizada sempre que um arquivo é gravado.

`auto` e `noauto`

Se o sistema de arquivos pode (ou não) ser montado automaticamente com `mount -a`.

`defaults`

Passa as opções `rw`, `suid`, `dev`, `exec`, `auto`, `nouser` e `async` para `mount`.

`dev` e `nodev`

Indica se os dispositivos de caractere ou de bloco no sistema de arquivos montado devem ser interpretados.

`exec` e `noexec`

Permite ou nega a permissão para executar binários no sistema de arquivos.

`user` e `nouser`

Permite (ou não) a um usuário comum montar o sistema de arquivos.

`group`

Permite a um usuário montar o sistema de arquivos se o usuário pertencer ao mesmo grupo que possui o dispositivo que o contém.

`owner`

Permite a um usuário montar um sistema de arquivos se ele for proprietário do dispositivo que o contém.

`suid` e `nosuid`

Permite ou não que os bits SETUID e SETGID tenham efeito.

`ro` e `rw`

Montam um sistema de arquivos como somente leitura ou gravável.

`remount`

Tenta remontar um sistema de arquivos já montado. Não é usado em `/etc/fstab`, mas como um parâmetro para `mount -o`. Por exemplo, para remontar a partição já montada `/dev/sdb1` como somente leitura, você pode usar o comando `mount -o remount,ro /dev/sdb1`. Ao remontar, não é necessário especificar o tipo de sistema de arquivos, apenas o nome do dispositivo *ou* o ponto de montagem.

`sync` e `async`

Definem se todas as operações de E/S devem ser realizadas no sistema de arquivos de forma síncrona ou assíncrona. `async` geralmente é o padrão. A página de manual de `mount` avisa que usar `sync` em mídias com um número limitado de ciclos de gravação (como drives flash ou cartões de memória) pode encurtar a vida útil do dispositivo.

Usando UUIDs e rótulos

Podem ocorrer problemas ao se especificar o nome do dispositivo que contém o sistema de arquivos a ser montado. Às vezes, o mesmo nome pode ser atribuído a outro dispositivo, dependendo de quando ou onde ele foi conectado ao sistema. Por exemplo, um pendrive em `/dev/sdb1` pode ser atribuído a `/dev/sdc1` se conectado em outra porta ou após outro pendrive.

Uma maneira de evitar isso é especificar o rótulo ou UUID (*Universally Unique Identifier*) do volume. Ambos são especificados quando o sistema de arquivos é criado e não serão alterados, a menos que o sistema de arquivos seja destruído ou atribuído manualmente a um novo rótulo ou UUID.

O comando `lsblk` serve para consultar informações sobre um sistema de arquivos e descobrir o rótulo e o UUID associados a ele. Para isso, use o parâmetro `-f`, seguido pelo nome do dispositivo:

```
$ lsblk -f /dev/sda1
NAME FSTYPE LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sda1 ext4         6e2c12e3-472d-4bac-a257-c49ac07f3761   64,9G    33% /
```

Este é o significado de cada coluna:

`NAME`
: Nome do dispositivo que contém o sistema de arquivos.

`FSTYPE`
: Tipo de sistema de arquivos.

`LABEL`
: Rótulo do sistema de arquivos.

`UUID`
: Identificador Universal Único (UUID) atribuído ao sistema de arquivos.

`FSAVAIL`
: Espaço disponível no sistema de arquivos.

`FSUSE%`
: Porcentagem de uso do sistema de arquivos.

`MOUNTPOINT`
: Onde o sistema de arquivos é montado.

Em `/etc/fstab`, um dispositivo pode ser especificado por seu UUID com a opção `UUID=` seguida pelo UUID, ou com `LABEL=`, seguida pelo rótulo. Assim, em vez de:

```
/dev/sda1  /  ext4  noatime,errors
```

Usaríamos:

```
UUID=6e2c12e3-472d-4bac-a257-c49ac07f3761  /  ext4  noatime,errors
```

Ou, se tivermos um disco rotulado como `homedisk`:

```
LABEL=homedisk  /home ext4  defaults
```

A mesma sintaxe pode ser usada com o comando `mount`. Em vez do nome do dispositivo, passe o UUID ou rótulo. Por exemplo, para montar um disco NTFS externo com o UUID `56C11DCC5D2E1334` em `/mnt/external`, o comando seria:

```
$ mount -t ntfs UUID=56C11DCC5D2E1334 /mnt/external
```

Montando discos com Systemd

Systemd é o processo init, o primeiro processo a ser executado em muitas distribuições Linux. Ele é responsável por gerar outros processos, iniciar serviços e inicializar o sistema. Entre muitas outras tarefas, o systemd também pode ser usado para gerenciar a montagem (e a montagem automática) de sistemas de arquivos.

Para usar este recurso do systemd, você precisa criar um arquivo de configuração chamado *unidade de montagem*. Cada volume a ser montado tem sua própria unidade de montagem e elas devem ser postas em `/etc/systemd/system/`.

As unidades de montagem são arquivos de texto simples com a extensão `.mount`. O formato básico é mostrado abaixo:

```
[Unit]
Description=

[Mount]
What=
```

```
Where=
Type=
Options=

[Install]
WantedBy=
```

`Description=`

Descrição curta da unidade de montagem, algo como `Mounts the backup disk`.

`What=`

O que deve ser montado. O volume tem de ser especificado como `/dev/disk/by-uuid/VOL_UUID`, onde `VOL_UUID` é o UUID do volume.

`Where=`

Deve ser o caminho completo para o local em que o volume será montado.

`Type=`

O tipo de sistema de arquivos.

`Options=`

Opções de montagem que podem ser desejáveis; são as mesmas usadas com o comando `mount` ou em `/etc/fstab`.

`WantedBy=`

Usado para o gerenciamento de dependências. Neste caso, usaremos `multi-user.target`, que indica que sempre que o sistema inicializar em um ambiente multiusuário (uma inicialização normal) a unidade será montada.

Nosso exemplo anterior do disco externo poderia ser escrito como:

```
[Unit]
Description=External data disk

[Mount]
What=/dev/disk/by-uuid/56C11DCC5D2E1334
Where=/mnt/external
Type=ntfs
Options=defaults

[Install]
```

```
WantedBy=multi-user.target
```

Mas ainda não acabamos. Para funcionar corretamente, a unidade de montagem *deve* ter o mesmo nome do ponto de montagem. Neste caso, o ponto de montagem é `/mnt/external`, de forma que o nome do arquivo precisa ser `mnt-external.mount`.

Depois disso, precisamos reiniciar o daemon do systemd com o comando `systemctl` e iniciar a unidade:

```
# systemctl daemon-reload
# systemctl start mnt-external.mount
```

Agora o conteúdo do disco externo deve estar disponível em `/mnt/external`. Para verificar o status da montagem, use o comando `systemctl status mnt-external.mount`, como mostrado abaixo:

```
# systemctl status mnt-external.mount
● mnt-external.mount - External data disk
   Loaded: loaded (/etc/systemd/system/mnt-external.mount; disabled; vendor pres
   Active: active (mounted) since Mon 2019-08-19 22:27:02 -03; 14s ago
    Where: /mnt/external
     What: /dev/sdb1
    Tasks: 0 (limit: 4915)
   Memory: 128.0K
   CGroup: /system.slice/mnt-external.mount

ago 19 22:27:02 pop-os systemd[1]: Mounting External data disk...
ago 19 22:27:02 pop-os systemd[1]: Mounted External data disk.
```

O comando `systemctl start mnt-external.mount` só habilita a unidade para a sessão atual. Se quiser habilitá-la em todas as inicializações, substitua `start` por `enable`:

```
# systemctl enable mnt-external.mount
```

Montagem automática de uma unidade de montagem

As unidades de montagem podem ser montadas automaticamente sempre que o ponto de montagem for acessado. Para isso, precisamos de um arquivo `.automount`, junto com o arquivo `.mount` descrevendo a unidade. O formato básico é:

```
[Unit]
Description=

[Automount]
Where=

[Install]
WantedBy=multi-user.target
```

Como anteriormente, `Description=` é uma breve descrição do arquivo e `Where=` é o ponto de montagem. Por exemplo, um arquivo `.automount` em nosso exemplo anterior ficaria assim:

```
[Unit]
Description=Automount for the external data disk

[Automount]
Where=/mnt/external

[Install]
WantedBy=multi-user.target
```

Salve o arquivo com o mesmo nome do ponto de montagem (neste caso, `mnt-external.automount`), recarregue o systemd e inicie a unidade:

```
# systemctl daemon-reload
# systemctl start mnt-external.automount
```

Agora, sempre que o diretório `/mnt/external` for acessado, o disco será montado. Como anteriormente, para habilitar a montagem automática em cada inicialização usaríamos:

```
# systemctl enable mnt-external.automount
```

Exercícios Guiados

1. Usando `mount`, como você montaria um sistema de arquivos `ext4` em `/dev/sdc1` para `/mnt/external` como somente leitura, usando as opções `noatime` e `async`?

2. Ao desmontar um sistema de arquivos em `/dev/sdd2`, aparece a mensagem de erro `target is busy`. Como descobrir quais arquivos do sistema de arquivos estão abertos e quais processos os abriram?

3. Considere a seguinte entrada em `/etc/fstab`: `/dev/sdb1 /data ext4 noatime,noauto,async`. Esse sistema de arquivos será montado se o comando `mount -a` for emitido? Por quê?

4. Como descobrir o UUID de um sistema de arquivos sob `/dev/sdb1`?

5. Como usar `mount` para remontar como somente leitura um sistema de arquivos exFAT com o UUID `6e2c12e3-472d-4bac-a257-c49ac07f3761`, montado em `/mnt/data`?

6. Como obter uma lista de todos os sistemas de arquivos `ext3` e `ntfs` atualmente montados em um sistema?

Exercícios Exploratórios

1. Considere a entrada a seguir em `/etc/fstab`: `/dev/sdc1 /backup ext4 noatime,nouser,async`. Seria possível um usuário montar esse sistema de arquivos com o comando `mount /backup`? Por quê?

2. Considere um sistema de arquivos remoto montado em `/mnt/server` que se tornou inacessível devido à perda de conectividade de rede. Como poderíamos forçar a desmontagem, ou montá-lo como somente leitura se isso não for possível?

3. Escreva uma entrada de `/etc/fstab` para montar um volume btrfs com o rótulo `Backup` em `/mnt/backup`, com opções padrão e sem permitir a execução de binários por ele.

4. Considere a seguinte unidade de montagem do systemd:

```
[Unit]
Description=External data disk

[Mount]
What=/dev/disk/by-uuid/56C11DCC5D2E1334
Where=/mnt/external
Type=ntfs
Options=defaults

[Install]
WantedBy=multi-user.target
```

   - Qual seria uma entrada equivalente a `/etc/fstab` neste sistema de arquivos?

5. Qual deve ser o nome de arquivo da unidade acima, para que ela possa ser usada pelo systemd? Onde ela deve ser posta?

Resumo

Nesta lição, você aprendeu a montar e desmontar sistemas de arquivos, manual ou automaticamente. Alguns dos comandos e conceitos explicados foram:

- `mount` (monta um dispositivo em um local)
- `umount` (desmonta um dispositivo)
- `lsof` (lista os processos que acessam um sistema de arquivos)
- Diretórios `/mnt` e `/media`
- `/etc/fstab`
- `lsblk` (lista o tipo e o UUID de um sistema de arquivos)
- Como montar um sistema de arquivos usando seu UUID ou rótulo.
- Como montar um sistema de arquivos usando unidades de montagem do systemd.
- Como montar automaticamente um sistema de arquivos usando unidades de montagem do systemd.

Respostas aos Exercícios Guiados

1. Usando `mount`, como você montaria um sistema de arquivos `ext4` em `/dev/sdc1` para `/mnt/external` como somente leitura, usando as opções `noatime` e `async`?

   ```
   # mount -t ext4 -o noatime,async,ro /dev/sdc1 /mnt/external
   ```

2. Ao desmontar um sistema de arquivos em `/dev/sdd2`, aparece a mensagem de erro `target is busy`. Como descobrir quais arquivos do sistema de arquivos estão abertos e quais processos os abriram?

   Use `lsof` seguido pelo nome do dispositivo:

   ```
   $ lsof /dev/sdd2
   ```

3. Considere a seguinte entrada em `/etc/fstab`: `/dev/sdb1 /data ext4 noatime,noauto,async`. Esse sistema de arquivos será montado se o comando `mount -a` for emitido? Por quê?

   Ele não será montado. A chave é o parâmetro `noauto`, que indica que esta entrada será ignorada por `mount -a`.

4. Como descobrir o UUID de um sistema de arquivos sob `/dev/sdb1`?

   Use `lsblk -f`, seguido pelo nome do sistema de arquivos:

   ```
   $ lsblk -f /dev/sdb1
   ```

5. Como usar `mount` para remontar como somente leitura um sistema de arquivos exFAT com o UUID `6e2c12e3-472d-4bac-a257-c49ac07f3761`, montado em `/mnt/data`?

   Visto que o sistema de arquivos está montado, você não precisa se preocupar com o tipo de sistema de arquivos ou o ID, basta usar a opção `remount` com o parâmetro `ro` (somente leitura) e o ponto de montagem:

   ```
   # mount -o remount,ro /mnt/data
   ```

6. Como obter uma lista de todos os sistemas de arquivos `ext3` e `ntfs` atualmente montados em um sistema?

Use `mount -t`, seguido por uma lista separada por vírgula dos sistemas de arquivos:

```
# mount -t ext3,ntfs
```

Respostas aos Exercícios Exploratórios

1. Considere a entrada a seguir em `/etc/fstab`: `/dev/sdc1 /backup ext4 noatime,nouser,async`. Seria possível um usuário montar esse sistema de arquivos com o comando `mount /backup`? Por quê?

   Não, o parâmetro `nouser` impede que usuários comuns montem este sistema de arquivos.

2. Considere um sistema de arquivos remoto montado em `/mnt/server` que se tornou inacessível devido à perda de conectividade de rede. Como poderíamos forçar a desmontagem, ou montá-lo como somente leitura se isso não for possível?

   Passe os parâmetros `-f` e `-r` para unmount. O comando seria `umount -f -r /mnt/server`. Lembre-se de que podemos agrupar parâmetros, então `umount -fr /mnt/server` também funcionaria.

3. Escreva uma entrada de `/etc/fstab` para montar um volume btrfs com o rótulo `Backup` em `/mnt/backup`, com opções padrão e sem permitir a execução de binários por ele.

   A linha seria `LABEL=Backup /mnt/backup btrfs defaults,noexec`

4. Considere a seguinte unidade de montagem do systemd:

   ```
   [Unit]
   Description=External data disk

   [Mount]
   What=/dev/disk/by-uuid/56C11DCC5D2E1334
   Where=/mnt/external
   Type=ntfs
   Options=defaults

   [Install]
   WantedBy=multi-user.target
   ```

   - Qual seria uma entrada equivalente a `/etc/fstab` neste sistema de arquivos??

     A entrada seria: `UUID=56C11DCC5D2E1334 /mnt/external ntfs defaults`

5. Qual deve ser o nome de arquivo da unidade acima, para que ela possa ser usada pelo systemd? Onde ela deve ser posta?

O nome de arquivo deve ser idêntico ao do ponto de montagem, no caso `mnt-external.mount`, posto em `/etc/systemd/system`.

104.5 Controlar permissões e propriedades de arquivos

Referência ao LPI objectivo

LPIC-1 version 5.0, Exam 101, Objective 104.5

Peso

3

Áreas chave de conhecimento

- Gerenciar permissões de acesso a arquivos comuns e especiais, bem como aos diretórios.
- Usar os modos de acesso tais como suid, sgid e o sticky bit (bit de aderência) para manter a segurança.
- Saber como mudar a máscara de criação de arquivo.
- Usar o campo de grupo para conceder acesso para grupos de trabalho.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `chmod`
- `umask`
- `chown`
- `chgrp`

104.5 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	104 Dispositivos, sistemas de arquivos do Linux, hierarquia padrão de sistemas de arquivos
Objetivo:	104.5 Gerenciar permissões e propriedade de arquivos
Lição:	1 de 1

Introdução

Por ser um sistema multiusuário, o Linux precisa de alguma forma de rastrear quem é o proprietário de cada arquivo e se um usuário tem ou não permissão para executar ações em um arquivo. Isso serve para garantir a privacidade dos usuários que desejam manter o conteúdo de seus arquivos em sigilo, bem como para permitir colaboração, tornando certos arquivos acessíveis a diversos usuários.

Isso é feito por meio de um sistema de permissões em três níveis. Cada arquivo em disco pertence a um usuário e a um grupo de usuários e tem três conjuntos de permissões: um para seu proprietário, um para o grupo que possui o arquivo e um para todos os outros. Nesta lição, você aprenderá a consultar as permissões de um arquivo, o significado dessas permissões e como manipulá-las.

Consulta de informações sobre arquivos e diretórios

O comando `ls` é usado para obter uma lista do conteúdo de qualquer diretório. Em sua forma mais básica, tudo o que você obtém são os nomes dos arquivos:

```
$ ls
Another_Directory  picture.jpg  text.txt
```

Mas há muito mais informações disponíveis para cada arquivo, incluindo seu tipo, tamanho, propriedade e muito mais. Para ver essas informações, você deve pedir ao `ls` uma lista de "formato longo", usando o parâmetro `-l`:

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Cada coluna da saída acima tem um significado. Vamos dar uma olhada nas colunas relevantes para esta lição.

- A *primeira* coluna da lista mostra o tipo de arquivo e as permissões. Por exemplo, em `drwxrwxr-x`:
  - O primeiro caractere, `d`, indica o tipo de arquivo.
  - Os três caracteres seguintes, `rwx`, indicam as permissõe do proprietário do arquivo, também chamado de *usuário* ou `u`.
  - Os três caracteres seguintes, `rwx`, indicam as permissões do *grupo* que possui o arquivo, também chamado de `g`.
  - Os três últimos caracteres, `r-x`, indicam as permissões de todos os *outros* ou `o`.

TIP Também é comum chamar as permissões de *outros* de permissões *world* (mundo), como em "Todo mundo tem essas permissões".

- A *terceira* e *quarta* colunas mostram informações sobre a propriedade: respectivamente o usuário e o grupo que possuem o arquivo.
- A *sétima* e a última colunas mostram o nome do arquivo.

A *segunda* coluna indica o número de links físicos que apontam para aquele arquivo. A *quinta* coluna mostra o tamanho do arquivo. A *sexta* coluna mostra a data e hora em que o arquivo foi modificado pela última vez. Mas essas colunas não são relevantes para o tópico atual.

E quanto aos diretórios?

Se você solicitar informações sobre um diretório usando `ls -l`, ele mostra uma lista do conteúdo do diretório:

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

Para evitar isso e consultar informações sobre o próprio diretório, adicione o parâmetro `-d` a `ls`:

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

Exibindo arquivos ocultos

A listagem do diretório que recuperamos usando `ls -l` anteriormente está incompleta:

```
$ ls -l
total 544
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Existem três outros arquivos nesse diretório, mas eles estão ocultos. No Linux, os arquivos cujo nome começa com um ponto (`.`) são ocultados automaticamente. Para vê-los, precisamos adicionar o parâmetro `-a` ao `ls`:

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

O arquivo `.thisIsHidden` está oculto simplesmente porque o nome começa com `.`.

Os diretórios `.` e `..`, porém, são especiais. `.` é um ponteiro para o diretório atual. E `..` é um ponteiro para o diretório pai, aquele que contém o atual. No Linux, cada diretório contém pelo menos esses dois diretórios.

TIP É possível combinar os parâmetros do `ls` (e muitos outros comandos do Linux). `ls -l -a`, por exemplo, pode ser escrito como `ls -la`.

Entendendo os tipos de arquivos

Já mencionamos que a primeira letra em cada saída de `ls -l` descreve o tipo do arquivo. Os três tipos de arquivo mais comuns são:

`-` (arquivo normal)

Um arquivo pode conter dados de qualquer tipo e ajuda a gerenciar esses dados. Os arquivos podem ser modificados, movidos, copiados e excluídos.

`d` (diretório)

Um diretório contém outros arquivos ou diretórios e ajuda a organizar o sistema de arquivos. Tecnicamente, os diretórios são um tipo especial de arquivo.

`l` (link simbólico)

Este "arquivo" é um ponteiro para outro arquivo ou diretório em outro local no sistema de arquivos.

Além desses, existem três outros tipos de arquivo que você precisa pelo menos conhecer, mas estão fora do escopo desta lição:

`b` (dispositivo de bloco)

Este arquivo representa um dispositivo virtual ou físico, geralmente discos ou outros tipos de dispositivos de armazenamento, como o primeiro disco rígido, que pode ser representado por `/dev/sda`.

`c` (dispositivo de caracteres)

Este arquivo representa um dispositivo virtual ou físico. Terminais (como o terminal principal em `/dev/ttyS0`) e portas seriais são exemplos comuns de dispositivos de caracteres.

`s` (socket)

Sockets servem como "canais" passando informações entre dois programas.

WARNING Não altere nenhuma permissão nos dispositivos de bloco, dispositivos de

> caracteres ou sockets, a menos que saiba muito bem o que está fazendo. Isso pode fazer o sistema parar de funcionar!

Entendendo as permissões

Na saída de `ls -l`, as permissões de arquivo são mostradas logo após o tipo de arquivo, como três grupos de três caracteres cada, na ordem `r`, `w` e `x`. Eis o que significam. Lembre-se de que um traço `-` representa a falta de permissão.

Permissões de arquivos

`r`

Significa *read* (leitura) e tem um valor octal de `4` (não se preocupe, falaremos de octais em breve). Indica permissão para abrir um arquivo e ler seu conteúdo.

`w`

Significa *write* (escrita) e tem um valor octal de `2`. Indica permissão para editar ou excluir um arquivo.

`x`

Significa *execute* (execução) e tem um valor octal de `1`. Indica que o arquivo pode ser executado como um executável ou script.

Assim, por exemplo, um arquivo com permissões `rw-` pode ser lido e escrito, mas não pode ser executado.

Permissões em diretórios

`r`

Significa *read* (leitura) e tem um valor octal de `4`. Indica permissão para ler o conteúdo do diretório, como nomes de arquivos. Mas *não* implica em permissão para ler os arquivos em si.

`w`

Significa *write* (escrita) e tem um valor octal de `2`. Indica permissão para editar ou excluir arquivos em um diretório, ou alterar seus nomes, permissões e proprietários.

Se um usuário tiver a permissão `w` em um diretório, ele poderá alterar as permissões de qualquer arquivo dentro do diretório (o *conteúdo* do diretório), mesmo que o usuário não tenha permissões no arquivo ou se o arquivo pertencer a outro utilizador.

Lembre-se de que ter permissões de gravação em um diretório ou arquivo não significa que você

tem permissão para remover ou renomear o diretório ou arquivo em si.

x

Significa *execute* (execução) e tem um valor octal de 1. Indica permissão para entrar em um diretório, mas não para listar seus arquivos (para isso, r é necessário).

A última parte sobre diretórios pode parecer um pouco confusa. Vamos imaginar, por exemplo, que você tem um diretório chamado Another_Directory, com as seguintes permissões:

```
$ ls -ld Another_Directory/
d--x--x--x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Imagine também que dentro deste diretório há um script de shell chamado hello.sh:

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

Se você for a usuária carol e tentar listar o conteúdo de Another_Directory, receberá uma mensagem de erro, pois seu usuário não tem permissão de leitura para esse diretório:

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

No entanto, a usuária carol *tem* permissões de execução, o que significa que ela pode entrar no diretório. Portanto, a usuária carol pode acessar arquivos dentro do diretório, desde que tenha as permissões corretas *para o respectivo arquivo*. Vamos supor que a usuária tem permissões totais (rwx) para o script hello.sh. Nesse caso, se souber o nome do arquivo completo, ela *pode* executar o script, embora *não possa* ler o conteúdo do diretório que o contém:

```
$ sh Another_Directory/hello.sh
Hello LPI World!
```

Como dissemos antes, as permissões são especificadas em sequência: primeiro para o proprietário do arquivo, depois para o grupo proprietário e, em seguida, para outros usuários. Sempre que alguém tenta realizar uma ação no arquivo, as permissões são verificadas na mesma ordem.

Primeiro, o sistema verifica se o usuário atual possui o arquivo e, se for o caso, ele aplica apenas o primeiro conjunto de permissões. Caso contrário, ele verifica se o usuário atual pertence ao grupo que possui o arquivo. Nesse caso, ele aplica o segundo conjunto de permissões apenas. Em qualquer

outro caso, o sistema aplicará o terceiro conjunto de permissões.

Isso significa que, se o usuário atual for o proprietário do arquivo, apenas as permissões do proprietário serão efetivas, mesmo se as permissões do grupo ou outras forem mais permissivas do que as do proprietário.

Modificando as permissões de arquivos

O comando `chmod` é usado para modificar as permissões de um arquivo, e pede pelo menos dois parâmetros: o primeiro descreve quais permissões alterar, o segundo aponta para o arquivo ou diretório onde a alteração será feita. Lembre-se de que apenas o proprietário do arquivo ou o administrador do sistema (root) pode alterar as permissões em um arquivo.

As permissões a alterar podem ser descritas de duas maneiras, ou "modos", diferentes.

O primeiro, denominado *modo simbólico*, oferece um controle refinado, permitindo adicionar ou revogar uma única permissão sem modificar as outras no conjunto. O outro modo, chamado *modo octal*, é mais fácil de lembrar e mais rápido de usar quando desejamos definir todos os valores de permissão de uma vez.

Ambos os modos levam ao mesmo resultado final. Assim, por exemplo, os comandos:

```
$ chmod ug+rw-x,o-rwx text.txt
```

e

```
$ chmod 660 text.txt
```

produzem exatamente a mesma saída, um arquivo com as permissões definidas:

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

Agora, vamos ver como cada modo funciona.

Modo simbólico

Ao descrever quais permissões alterar no *modo simbólico*, o(s) primeiro(s) caractere(s) indica(m) as permissões que serão alteradas: de usuário (`u`), grupo (`g`), outros (`o`) e/ou todos (`a`).

Então você precisa dizer ao comando o que fazer: você pode conceder uma permissão (`+`), revogar uma permissão (`-`) ou defini-la com um valor específico (`=`).

Por último, você especifica em qual permissão deseja agir: leitura (`r`), escrita (`w`) ou execução (`x`).

Por exemplo, imagine que temos um arquivo chamado `text.txt` com o seguinte conjunto de permissões:

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Se você deseja conceder permissões de gravação aos membros do grupo proprietário do arquivo, deve usar o parâmetro `g+w`. É mais fácil pensar desta forma: "Para o grupo (`g`), conceda (`+`) permissões de escrita (`w`)". Então, o comando seria:

```
$ chmod g+w text.txt
```

Vamos conferir o resultado com `ls`:

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Deseja remover as permissões de leitura para o proprietário do mesmo arquivo? Pense assim: "Para o usuário (`u`), revogue (`-`) as permissões de leitura (`r`)". Portanto, o parâmetro é `u-r`, desta maneira:

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

E se quisermos definir as permissões exatamente como `rw-` para todos? Nesse caso, pensamos assim: "Para todos (`a`), defina exatamente (`=`) leitura (`r`), escrita (`w`), e não execução (`-`)". Assim:

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Claro, é possível modificar diversas permissões ao mesmo tempo. Neste caso, separe-os com uma

vírgula (`,`):

```
$ chmod u+rwx,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

O exemplo acima pode ser lido como: "Para o usuário (`u`), conceda (`+`) permissões de leitura, escrita e execução (`rwx`), para o grupo (`g`), revogue (`-`) permissões de execução (`x`)".

Quando executado em um diretório, `chmod` modifica apenas as permissões do diretório. `chmod` também possui um modo recursivo, útil quando desejamos alterar as permissões para "todos os arquivos dentro de um diretório e seus subdiretórios". Para usá-lo, adicione o parâmetro `-R` após o nome do comando, antes das permissões a alterar:

```
$ chmod -R u+rwx Another_Directory/
```

Este comando pode ser lido como: "Recursivamente (`-R`), para o usuário (`u`), conceda (`+`) permissões de leitura, escrita e execução (`rwx`)".

WARNING

Tenha cuidado e pense duas vezes antes de usar a opção `-R`, pois é fácil alterar sem querer as permissões de arquivos e diretórios, especialmente em diretórios com um grande número de arquivos e subdiretórios.

Modo octal

No *modo octal*, as permissões são especificadas de maneira diferente: como um valor de três dígitos na notação octal, um sistema numérico de base 8.

Cada permissão tem um valor correspondente e elas são especificadas na seguinte ordem: primeiro vem leitura (`r`), que é `4`, depois escrita (`w`), que é `2`, e por fim execução (`x`), representada por `1`. Se não houver permissão, usamos o valor zero (`0`). Portanto, uma permissão `rwx` seria `7` (`4 + 2 + 1`) e `rx` seria `5` (`4 + 0 + 1`).

O primeiro dos três dígitos no conjunto de permissões representa as permissões do usuário (`u`), o segundo as do grupo (`g`) e o terceiro as do outros (`o`). Se quisermos definir as permissões de um arquivo como `rw-rw----`, o valor octal seria `660`:

```
$ chmod 660 text.txt
$ ls -l text.txt
```

```
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Além disso, a sintaxe no *modo octal* é a mesma que no *modo simbólico*: o primeiro parâmetro representa as permissões que você deseja alterar e o segundo aponta para o arquivo ou diretório onde a alteração será feita.

TIP Se o valor de uma permissão for *ímpar*, o arquivo certamente é executável!

Qual sintaxe você deve usar? O *modo octal* é recomendado quando se deseja alterar as permissões para um valor específico, por exemplo `640` (`rw- r-- ---`).

O *modo simbólico* é mais útil se você deseja inverter apenas um valor específico, independentemente das permissões atuais do arquivo. Por exemplo, você pode adicionar permissões de execução para o usuário usando apenas `chmod u+x script.sh` sem levar em conta, ou mesmo tocar, as permissões atuais de grupo e outros.

Modificando o proprietário de um arquivo

O comando `chown` é usado para modificar a propriedade de um arquivo ou diretório. A sintaxe é bastante simples:

```
chown USERNAME:GROUPNAME FILENAME
```

Por exemplo, vamos verificar um arquivo chamado `text.txt`:

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

O usuário que possui o arquivo é `carol`, e o grupo também é `carol`. Agora, vamos mudar o grupo proprietário do arquivo para um outro grupo, como `students`:

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Lembre-se de que o usuário que possui um arquivo não precisa pertencer ao grupo que possui o arquivo. No exemplo acima, o usuário `carol` não precisa ser um membro do grupo `students`.

O conjunto de permissões de usuário ou grupo pode ser omitido se você não quiser alterá-lo.

Portanto, para alterar apenas o grupo que possui um arquivo, o comando seria `chown: students text.txt`. Para alterar apenas o usuário, o comando seria `chown carol: text.txt` ou apenas `chown carol text.txt`. Alternativamente, você pode usar o comando `chgrp students text.txt`.

A menos que você seja o administrador do sistema (root), não é possível mudar a propriedade de um arquivo para outro usuário ou grupo ao qual você não pertence. Se tentar fazer isso, aparecerá a mensagem de erro `Operation not allowed`.

Consultando os grupos

Antes de alterar a propriedade de um arquivo, pode ser útil saber quais grupos existem no sistema, quais usuários são membros de um grupo e a quais grupos um usuário pertence.

Para ver quais grupos existem em seu sistema, digite `getent group`. A saída será semelhante a esta (a saída foi abreviada):

```
$ getent group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,rigues
tty:x:5:rigues
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:rigues
```

Se quiser saber a quais grupos um usuário pertence, adicione o nome de usuário como um parâmetro a `groups`:

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

Para fazer o inverso (ver quais usuários pertencem a um grupo), use `groupmems`. O parâmetro `-g` especifica o grupo, e `-l` lista todos os seus membros:

```
# groupmems -g cdrom -l
carol
```

TIP `groupmems` só pode ser executado como root, o administrador do sistema. Se você não estiver conectado como root, adicione `sudo` antes do comando.

Permissões padrão

Vamos fazer uma experiência. Abra uma janela de terminal e crie um arquivo vazio com o seguinte comando:

```
$ touch testfile
```

Agora, vamos dar uma olhada nas permissões deste arquivo. Eles *podem* ser diferentes em seu sistema, mas vamos supor que sejam mais ou menos assim:

```
$ ls -lh testfile
-rw-r--r-- 1 carol carol 0 jul 13 21:55 testfile
```

As permissões são `rw-r--r--`: *leitura* e *escrita* para o usuário, e *leitura* para o grupo e outros, ou `644` no modo octal. Agora, tente criar um diretório:

```
$ mkdir testdir
$ ls -lhd testdir
drwxr-xr-x 2 carol carol 4,0K jul 13 22:01 testdir
```

Agora as permissões são `rwxr-xr-x`: *leitura*, *escrita* e *execução* para o usuário, *leitura* e *execução* para o grupo e outros, ou `755` no modo octal.

Não importa onde você esteja no sistema de arquivos, cada arquivo ou diretório que criar terá as mesmas permissões. Você já se perguntou de onde eles vêm?

Elas vêm da *máscara de usuário* ou `umask`, que define as permissões padrão para cada arquivo criado. Você pode verificar os valores atuais com o comando `umask`:

```
$ umask
0022
```

Mas isso não se parece com `rw-r--r--`, ou mesmo `644`. Talvez devêssemos tentar com o parâmetro `-S`, para obter uma saída em modo simbólico:

```
$ umask -S
u=rwx,g=rx,o=rx
```

Essas são as mesmas permissões que nosso diretório de teste obteve em um dos exemplos acima. Mas por que quando criamos um arquivo as permissões eram diferentes?

Bem, não faz sentido definir permissões globais de execução para todos em qualquer arquivo por padrão, certo? Os diretórios precisam de permissões de execução (caso contrário, não é possível entrar neles), mas os arquivos não, então eles não as recebem. Daí o `rw-r--r--`.

Além de exibir as permissões padrão, o `umask` também pode ser usado para alterá-las para sua sessão do shell atual. Por exemplo, se usarmos o comando:

```
$ umask u=rwx,g=rwx,o=
```

Cada novo diretório irá herdar as permissões `rwxrwx---`, e cada arquivo `rw-rw----` (já que eles não recebem permissões de execução). Se você repetir os exemplos acima para criar um `testfile` e `testdir` e verificar as permissões, o resultado será:

```
$ ls -lhd test*
drwxrwx--- 2 carol carol 4,0K jul 13 22:25 testdir
-rw-rw---- 1 carol carol    0 jul 13 22:25 testfile
```

E se você marcar o `umask` sem o parâmetro `-S` (modo simbólico), você obterá:

```
$ umask
0007
```

O resultado não parece familiar porque os valores usados são diferentes. Eis uma tabela com todos os valores e seus respectivos significados:

Value	Permission for Files	Permission for Directories
0	rw-	rwx
1	rw-	rw-
2	r--	r-x
3	r--	r--

Value	Permission for Files	Permission for Directories
4	`-w-`	`-wx`
5	`-w-`	`-w-`
6	`---`	`--x`
7	`---`	`---`

Como vemos, `007` corresponde a `rwxrwx---`, exatamente como solicitamos. O zero inicial pode ser ignorado.

Permissões especiais

Além das permissões de leitura, gravação e execução para usuário, grupo e outros, cada arquivo pode ter três outras *permissões especiais* capazes de alterar a maneira como um diretório funciona ou como um programa é executado. Elas podem ser especificadas no modo simbólico ou octal e são as seguintes:

Sticky Bit

O sticky bit, também chamado de *sinalizador de exclusão restrito*, tem o valor octal `1` e no modo simbólico é representado por um `t` dentro das permissões de outros. Ele se aplica apenas a diretórios e não tem efeito em arquivos normais. No Linux, ele evita que os usuários removam ou renomeiem um arquivo em um diretório, a menos que sejam proprietários desse arquivo ou diretório.

Os diretórios com o sticky bit definido mostram um `t` substituindo o `x` nas permissões de *outros* na saída de `ls -l`:

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

No modo octal, as permissões especiais são especificadas usando uma notação de 4 dígitos, sendo que o primeiro dígito representa a permissão especial sobre a qual agir. Por exemplo, para definir o sticky bit (valor `1`) para o diretório `Another_Directory` no modo octal, com permissões `755`, o comando seria:

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Set GID

O Set GID, também conhecido como SGID ou Set Group ID bit, tem o valor octal `2` e no modo simbólico é representado por um `s` nas permissões de *grupo*. Ele pode ser aplicado a arquivos executáveis ou diretórios. Nos arquivos, fará com que o processo seja executado com os privilégios do grupo que possui o arquivo. Quando aplicado a diretórios, fará com que cada arquivo ou diretório criado herde o grupo do diretório pai.

Arquivos e diretórios com o bit SGID mostram um `s` no lugar do `x` nas permissões de *grupo* na saída de `ls -l`:

```
$ ls -l test.sh
-rwxr-sr-x 1 carol root 33 Dec 11 10:36 test.sh
```

Para adicionar permissões SGID a um arquivo no modo simbólico, o comando seria:

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

O exemplo a seguir o ajudará a entender melhor os efeitos do SGID em um diretório. Suponha que temos um diretório chamado `Sample_Directory`, pertencente à usuária `carol` e ao grupo `users`, com a seguinte estrutura de permissões:

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

Agora, vamos mudar para esse diretório e, usando o comando `touch`, criar um arquivo vazio dentro dele. O resultado seria:

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

Como podemos ver, o arquivo é propriedade da usuária `carol` e do grupo `carol`. Mas, se o diretório tivesse a permissão SGID definida, o resultado seria diferente. Primeiro, vamos adicionar o bit SGID ao `Sample_Directory` e verificar os resultados:

```
$ sudo chmod g+s Sample_Directory/
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

O `s` nas permissões do grupo indica que o bit SGID está definido. Agora, vamos mudar para este diretório e, novamente, criar um arquivo vazio com o comando `touch`:

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
 -rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

O grupo que possui o arquivo é `users`. Isso ocorre porque o bit SGID fez o arquivo herdar o proprietário do grupo de seu diretório pai, que é `users`.

Set UID

SUID, também conhecido como Set User ID, tem valor octal `4` e é representado por um `s` nas permissões de *usuário* no modo simbólico. Aplica-se apenas aos arquivos e não tem efeito em diretórios. Seu comportamento é semelhante ao do bit SGID, mas o processo será executado com os privilégios do *usuário* proprietário do arquivo. Os arquivos com o bit SUID mostram um `s` no lugar do `x` nas permissões do usuário, na saída de `ls -l`:

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Podemos combinar diversas permissões especiais em um parâmetro somando-as. Assim, para definir o SGID (valor 2) e o SUID (valor 4) no modo octal para o script `test.sh` com permissões `755`, digite:

```
$ chmod 6755 test.sh
```

E o resultado seria:

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

TIP Se o seu terminal exibe cores, como é o caso da maioria atualmente, é fácil ver se essas

permissões especiais estão definidas olhando a saída de `ls -l`. Para o sticky bit, o nome do diretório pode ser mostrado em uma fonte preta com fundo azul. O mesmo se aplica aos arquivos com os bits SGID (fundo amarelo) e SUID (fundo vermelho). As cores podem ser diferentes dependendo da distribuição do Linux e das configurações do terminal que você usa.

Exercícios Guiados

1. Crie um diretório chamado `emptydir` usando o comando `mkdir emptydir`. Em seguida, usando `ls`, liste as permissões do diretório `emptydir`.

2. Crie um arquivo vazio chamado `emptyfile` com o comando `touch emptyfile`. Em seguida, usando `chmod` no modo simbólico, adicione permissões de execução ao proprietário do arquivo `emptyfile` e remova as permissões de gravação e execução para todos os outros. Faça isso usando apenas um comando `chmod`.

3. Quais seriam as permissões padrão de um arquivo se o valor de `umask` for definido como `027`?

4. Vamos supor que um arquivo chamado `test.sh` seja um script do shell com as seguintes permissões e propriedade:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - Quais são as permissões do proprietário do arquivo?

   - Usando a notação octal, qual seria a sintaxe do `chmod` para "desfazer" a permissão especial concedida a este arquivo?

5. Considere este arquivo:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   Que tipo de arquivo é `sdb1`? E quem pode escrever nele?

6. Considere os 4 arquivos a seguir:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users  4,0K Jan 18 17:26 Sample_Directory
```

Escreva as permissões correspondentes a cada arquivo e diretório usando a notação numérica de 4 dígitos.

`Another_Directory`	
`foo.bar`	
`HugeFile.zip`	
`Sample_Directory`	

Exercícios Exploratórios

1. Experimente o seguinte em um terminal: crie um arquivo vazio chamado `emptyfile` com o comando `touch emptyfile`. Agora "zere" as permissões do arquivo com `chmod 000 emptyfile`. O que acontecerá se você mudar as permissões de `emptyfile` passando apenas *um* valor de `chmod` para o modo octal, como em `chmod 4 emptyfile`? E se usarmos dois, como em `chmod 44 emptyfile`? O que aprendemos sobre a maneira como `chmod` interpreta o valor numérico?

2. Considere as permissões do diretório temporário em um sistema Linux, `/tmp`:

```
$ ls -l /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

   Usuário, grupo e outros têm permissões totais. Mas um usuário regular pode excluir *qualquer* arquivo dentro deste diretório? Por quê?

3. Um arquivo chamado `test.sh` tem as seguintes permissões: `-rwsr-xr-x`, o que indica que o bit SUID foi definido. Agora, execute os seguintes comandos:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

   O que nós fizemos? O que significa o `S` maiúsculo?

4. Como criar um diretório chamado `Box` no qual todos os arquivos pertençam automaticamente ao grupo `users` e só possam ser removidos pelo usuário que os criou?

Resumo

Nesta lição, você aprendeu como usar o `ls` para obter (e decodificar) informações sobre permissões de arquivo, como controlar ou alterar quem pode criar, excluir ou modificar um arquivo com `chmod`, tanto nos modos *octal* e *simbólico* como para alterar a propriedade de arquivos com `chown` e `chgrp`, e como consultar e alterar a máscara de permissões padrão para arquivos e diretórios com `umask`.

Os seguintes comandos foram abordados nesta lição:

`ls`

Lista os arquivos, incluindo opcionalmente detalhes como as permissões.

`chmod`

Altera as permissões de um arquivo ou diretório.

`chown`

Altera o usuário e/ou o grupo proprietário de um arquivo ou diretório.

`chgrp`

Altera o grupo proprietário de um arquivo ou diretório.

`umask`

Permite consultar ou definir a máscara de permissões padrão para arquivos e diretórios

Respostas aos Exercícios Guiados

1. Crie um diretório chamado `emptydir` usando o comando `mkdir emptydir`. Em seguida, usando `ls`, liste as permissões do diretório `emptydir`.

   Adicionamos o parâmetro `-d` a `ls` para ver os atributos de arquivo de um diretório, em vez de listar seu conteúdo. Assim, a resposta é:

   ```
   ls -l -d emptydir
   ```

   Você ganha pontos extras se tiver juntado os dois parâmetros em um só, como em `ls -ld emptydir`.

2. Crie um arquivo vazio chamado `emptyfile` com o comando `touch emptyfile`. Em seguida, usando `chmod` no modo simbólico, adicione permissões de execução ao proprietário do arquivo `emptyfile` e remova as permissões de gravação e execução para todos os outros. Faça isso usando apenas um comando `chmod`.

   Pense desta maneira:

   - "Para o usuário que possui o arquivo (`u`) adicione (`+`) permissões de execução (`x`)", portanto `u+x`.
   - "Para o grupo (`g`) e outros usuários (`o`), remova (`-`) permissões de escrita (`w`) e execução (`x`)", portanto `go-wx`.

     Para combinar esses dois conjuntos de permissões, adicionamos uma virgula entre eles. Assim, o resultado final será:

     ```
     chmod u+x,go-wx emptyfile
     ```

3. Quais seriam as permissões padrão de um arquivo se o valor de `umask` for definido como `020`?

   As permissões seriam `rw-r-----`

4. Vamos supor que um arquivo chamado `test.sh` seja um script do shell com as seguintes permissões e propriedade:

   ```
   -rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
   ```

- Quais são as permissões do proprietário do arquivo?

  As permissões do proprietário (2º ao 4º caracteres na saída de `ls -l`) são `rwx`, por isso a resposta é: "ler, escrever e executar o arquivo".

- Usando a notação octal, qual seria a sintaxe do `chmod` para "desfazer" a permissão especial concedida a este arquivo?

  Podemos "desfazer" as permissões especiais passando um 4º dígito, `0`, para o `chmod`. As permissões atuais são `755`, de modo que o comando seria `chmod 0755`.

5. Considere este arquivo:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   Que tipo de arquivo é `sdb1`? E quem pode escrever nele?

   O primeiro caractere da saída de `ls -l` mostra o tipo de arquivo. `b` é um *dispositivo de bloco*, normalmente um disco (interno ou externo), conectado à máquina. O proprietário (`root`) e todos os usuários do grupo `disk` podem escrever nele.

6. Considere os 4 arquivos a seguir:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

   Escreva as permissões correspondentes a cada arquivo e diretório usando a notação numérica de 4 dígitos.

   As permissões correspondentes, em notação numérica, são as seguintes:

`Another_Directory`	`1755`. `1` para o sticky bit, `755` para as permissões regulares (`rwx` para o usuário, `r-x` para grupo e outros).

`foo.bar`	`0044`. Nenhuma permissão especial (por isso o primeiro dígito é `0`), sem permissões para o usuário (`---`) e somente leitura (`r--r--`) para grupo e outros.
`HugeFile.zip`	`0664`. Nenhuma permissão especial, por isso o primeiro dígito é `0`. `6` (`rw-`) para o usuário e grupo, `4` (`r--`) para os outros.
`Sample_Directory`	`2755`. `2` para o bit SGID, `7` (`rwx`) para o usuário, `5` (`r-x`) para o grupo e outros.

Respostas aos Exercícios Exploratórios

1. Experimente o seguinte em um terminal: crie um arquivo vazio chamado `emptyfile` com o comando `touch emptyfile`. Agora "zere" as permissões do arquivo com `chmod 000 emptyfile`. O que acontecerá se você mudar as permissões de `emptyfile` passando apenas *um* valor de `chmod` para o modo octal, como em `chmod 4 emptyfile`? E se usarmos dois, como em `chmod 44 emptyfile`? O que aprendemos sobre a maneira como `chmod` interpreta o valor numérico?

   Lembre-se de que "zeramos" as permissões de `emptyfile`. Assim, seu estado inicial seria:

   ```
   ---------- 1 carol carol    0 Dec 11 10:55 emptyfile
   ```

   Agora, vamos tentar o primeiro comando, `chmod 4 emptyfile`:

   ```
   $ chmod 4 emptyfile
   $ ls -l emptyfile
   -------r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Viu só? As permissões de *outros* foram alteradas. E se usássemos dois dígitos, como em `chmod 44 emptyfile`?

   ```
   $ chmod 44 emptyfile
   $ ls -l emptyfile
   ----r--r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Agora, as permissões de *grupo* e *outros* foram afetadas. A partir disso, concluímos que no modo octal o `chmod` lê o valor "de trás pra frente", do dígito menos significativo (*outros*) para o mais sigbificativo (*usuário*). Se você passar um dígito, modifica as permissões de *outros*. Com dois dígitos, modificamos *grupo* e *outros*, com três modificamos *usuário*, *grupo* e *outros* e com quatro dígitos modificamos *usuário*, *grupo*, *outros* e as permissões especiais.

2. Considere as permissões do diretório temporário em um sistema Linux, `/tmp`:

   ```
   $ ls -l /tmp
   drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
   ```

   Usuário, grupo e outros têm permissões totais. Mas um usuário regular pode excluir *qualquer* arquivo dentro deste diretório? Por quê?

`/tmp` é o que chamamos um *diretório escrito por todos*, ou seja, qualquer usuário pode escrever nele. Mas não queremos que um usuário modifique arquivos criados por outros, por isso o *sticky bit* foi definido (como indicado pelo `t` nas permissões de *outros*). Graças a isso, um usuário pode excluir arquivos em `/tmp`, mas somente se tiver criado esses arquivos.

3. Um arquivo chamado `test.sh` tem as seguintes permissões: `-rwsr-xr-x`, o que indica que o bit SUID foi definido. Agora, execute os seguintes comandos:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

   O que nós fizemos? O que significa o `S` maiúsculo?

   Removemos as permissões de execução do usuário dono do arquivo. O `s` (ou `t`) toma o lugar do `x` na saída de `ls -l`, por isso o sistema precisa de uma maneira de mostrar se o usuário tem permissões de execução ou não. Para isso, ele muda a caixa do caractere especial.

   Um `s` minúsculo no primeiro grupo de permissões indica que o usuário dono do arquivo tem permissões de execução e que o bit SUID foi definido. Um `S` maiúsculo indica que o usuário dono do arquivo não tem (–) permissões de execução e que o bit SUID foi definido.

   Pode-se dizer o mesmo do SGID. Um `s` minúsculo no segundo grupo de permissões indica que o grupo dono do arquivo tem permissões de execução e que o bit SUID foi definido. Um `S` maiúsculo indica que o grupo dono do arquivo não tem (–) permissões de execução e que o bit SUID foi definido.

   Isso também vale para o sticky bit, representado pelo `t` No terceiro grupo de permissões. O `t` minúsculo indica que o sticky bit foi definido e que os outros têm permissões de execução. O `T` maiúsculo indica que o sticky bit foi definido e que os outros não têm permissões de execução.

4. Como criar um diretório chamado `Box` no qual todos os arquivos pertençam automaticamente ao grupo `users` e só possam ser removidos pelo usuário que os criou?

   Este seria um processo de vários passos. O primeiro é criar o diretório:

```
$ mkdir Box
```

   Queremos que cada arquivo criado dentro deste diretório seja atribuído automaticamente ao grupo `users`. Para isso, configuramos esse grupo como proprietário do diretório, depois

definimos nele o bit SGID. Também precisamos garantir que qualquer membro do grupo possa escrever nesse diretório.

Como não precisamos nos importar com as outras permissões e queremos apenas "ativar" os bits especiais, faz sentido usar o modo simbólico:

```
$ chown :users Box/
$ chmod g+wxs Box/
```

Note que se seu usuário atual não pertencer ao grupo `users`, será necessário usar o comando `sudo` antes dos comandos acima para fazer a alteração como root.

Para terminar, vamos garantir que somente o usuário criador de um arquivo tem a permissão de excluí-lo. Para isso, definimos o sticky bit (representado por um `t`) no diretório. Lembre-se de que ele é configurado nas permissões de outros (`o`).

```
$ chmod o+t Box/
```

As permissões do diretório `Box` devem aparecer da seguinte maneira:

```
drwxrwsr-t 2 carol users  4,0K Jan 18 19:09 Box
```

Claro que também é possível especificar o bit SGID e o sticky bit usando somente um comando `chmod`:

```
$ chmod g+wxs,o+t Box/
```

Se você pensou nisso, parabéns!