



Linux
Professional
Institute

LPIC-1

Versão 5.0
Português

1001

Table of Contents

TÓPICO 101: ARQUITETURA DE SISTEMA	1
101.1 Identificar e editar configurações de hardware	2
101.1 Lição 1	3
Introdução	3
Ativação do dispositivo	4
Inspeção de dispositivos no Linux	4
Arquivos de informação e de dispositivo	11
Dispositivos de armazenamento	13
Exercícios Guiados	14
Exercícios Exploratórios	15
Resumo	16
Respostas aos Exercícios Guiados	17
Respostas aos Exercícios Exploratórios	18
101.2 Início (boot) do sistema	19
101.2 Lição 1	21
Introdução	21
BIOS ou UEFI	22
O bootloader	23
Inicialização do sistema	25
Inspeção da inicialização	26
Exercícios Guiados	30
Exercícios Exploratórios	31
Resumo	32
Respostas aos Exercícios Guiados	33
Respostas aos Exercícios Exploratórios	34
101.3 Alternar runlevels/boot targets, desligar e reiniciar o sistema	35
101.3 Lição 1	37
Introdução	37
SysVinit	38
systemd	41
Upstart	44
Desligar e reiniciar	46
Exercícios Guiados	48
Exercícios Exploratórios	49
Resumo	50
Respostas aos Exercícios Guiados	51
Respostas aos Exercícios Exploratórios	52
TÓPICO 102: INSTALAÇÃO DO LINUX E ADMINISTRAÇÃO DE PACOTES	53

102.1 Dimensionar partições de disco	54
102.1 Lição 1	55
Introdução	55
Pontos de montagem	56
Mantendo as coisas separadas	57
Swap	59
LVM	60
Exercícios Guiados	62
Exercícios Exploratórios	63
Resumo	64
Respostas aos Exercícios Guiados	65
Respostas aos Exercícios Exploratórios	66
102.2 Instalar o gerenciador de inicialização	67
102.2 Lição 1	68
Introdução	68
GRUB Legacy e GRUB 2	69
Onde fica o gerenciador de inicialização?	69
A partição /boot	70
GRUB 2	71
GRUB Legacy	78
Exercícios Guiados	82
Exercícios Exploratórios	83
Resumo	85
Respostas aos Exercícios Guiados	86
Respostas aos Exercícios Exploratórios	87
102.3 Controle de bibliotecas compartilhadas	89
102.3 Lição 1	90
Introdução	90
O que são bibliotecas compartilhadas	90
Convenções de nomenclatura para arquivos-objeto compartilhados	91
Configuração dos caminhos da biblioteca compartilhada	92
Buscando pelas dependências de um executável específico	95
Exercícios Guiados	97
Exercícios Exploratórios	98
Resumo	99
Respostas aos Exercícios Guiados	101
Respostas aos Exercícios Exploratórios	102
102.4 Utilização do sistema de pacotes Debian	103
102.4 Lição 1	104
Introdução	104

A ferramenta de pacotes do Debian (dpkg)	105
Ferramenta de pacotes avançada (apt).....	109
Exercícios Guiados.....	119
Exercícios Exploratórios	120
Resumo.....	121
Respostas aos Exercícios Guiados.....	123
Respostas aos Exercícios Exploratórios	124
102.5 Utilização do sistema de pacotes RPM e YUM	126
102.5 Lição 1	127
Introdução	127
O RPM Package Manager (rpm)	128
YellowDog Updater Modified (YUM).....	133
DNF	138
Zypper	140
Exercícios Guiados.....	147
Exercícios Exploratórios	148
Resumo.....	149
Respostas aos Exercícios Guiados.....	150
Respostas aos Exercícios Exploratórios	151
102.6 Linux virtualizado	152
102.6 Lição 1	153
Introdução	153
O que é virtualização?.....	153
Tipos de máquina virtual.....	154
Trabalhando com modelos de máquina virtual	161
Implementação de máquinas virtuais na nuvem	163
Contêiners	165
Exercícios Guiados.....	167
Exercícios Exploratórios	168
Resumo.....	169
Respostas aos Exercícios Guiados.....	170
Respostas aos Exercícios Exploratórios	171
TÓPICO 103: COMANDOS GNU E UNIX	173
103.1 Trabalhar na linha de comando	174
103.1 Lição 1	176
Introdução	176
Obtendo informações sobre o sistema.....	176
Obtendo informações sobre comandos	177
Usando o histórico de comandos	180
Exercícios Guiados.....	182

Exercícios Exploratórios	183
Resumo	184
Respostas aos Exercícios Guiados	185
Respostas aos Exercícios Exploratórios	186
103.1 Lição 2	187
Introdução	187
Encontrando suas variáveis de ambiente	187
Criando novas variáveis de ambiente	188
Removendo as variáveis de ambiente	189
Usando aspas para escapar dos caracteres especiais	190
Exercícios Guiados	192
Exercícios Exploratórios	193
Resumo	194
Respostas aos Exercícios Guiados	195
Respostas aos Exercícios Exploratórios	196
103.2 Processar fluxos de texto usando filtros	197
103.2 Lição 1	199
Introdução	199
Uma revisão rápida sobre redirecionamentos e pipes	199
Processando fluxos de texto	202
Exercícios Guiados	214
Exercícios Exploratórios	216
Resumo	218
Respostas aos Exercícios Guiados	221
Respostas aos Exercícios Exploratórios	226
103.3 Gerenciamento básico de arquivos	232
103.3 Lição 1	234
Introdução	234
Manipulação de arquivos	235
Criando e removendo diretórios	240
Manipulação recursiva de arquivos e diretórios	242
Globbing de arquivos e caracteres curinga	244
Tipos de caracteres curinga	245
Exercícios Guiados	249
Exercícios Exploratórios	251
Resumo	252
Respostas aos Exercícios Guiados	253
Respostas aos Exercícios Exploratórios	255
103.3 Lição 2	257
Introdução	257

Como encontrar arquivos	257
Arquivos de pacote	261
Exercícios Guiados.....	267
Exercícios Exploratórios	268
Resumo	269
Respostas aos Exercícios Guiados	270
Respostas aos Exercícios Exploratórios	271
103.4 Fluxos, pipes (canalização) e redirecionamentos de saída	273
103.4 Lição 1	274
Introdução	274
Redirecionamentos	275
Here Document e Here String	278
Exercícios Guiados.....	280
Exercícios Exploratórios	281
Resumo	282
Respostas aos Exercícios Guiados	283
Respostas aos Exercícios Exploratórios	284
103.4 Lição 2	285
Introdução	285
Pipes	285
Substituição de comando	287
Exercícios Guiados.....	290
Exercícios Exploratórios	291
Resumo	292
Respostas aos Exercícios Guiados	293
Respostas aos Exercícios Exploratórios	295
103.5 Criar, monitorar e finalizar processos	296
103.5 Lição 1	298
Introdução	298
Controle de jobs	298
Monitoramento de processos	303
Exercícios Guiados.....	315
Exercícios Exploratórios	317
Resumo	319
Respostas aos Exercícios Guiados	321
Respostas aos Exercícios Exploratórios	324
103.5 Lição 2	327
Introdução	327
Recursos dos multiplexadores de terminal	327
GNU Screen	328

tmux	335
Exercícios Guiados	344
Exercícios Exploratórios	347
Resumo	349
Respostas aos Exercícios Guiados	350
Respostas aos Exercícios Exploratórios	355
103.6 Modificar a prioridade de execução de um processo	357
103.6 Lição 1	358
Introdução	358
O Agendador do Linux	359
Como ler as prioridades	360
Valor nice	361
Exercícios Guiados	363
Exercícios Exploratórios	365
Resumo	366
Respostas aos Exercícios Guiados	367
Respostas aos Exercícios Exploratórios	369
103.7 Procurar em arquivos de texto usando expressões regulares	370
103.7 Lição 1	371
Introdução	371
Expressão de colchetes	372
Quantificadores	374
Chaves	374
Alternâncias e agrupamentos	375
Pesquisas com expressões regulares	375
Exercícios Guiados	377
Exercícios Exploratórios	378
Resumo	379
Respostas aos Exercícios Guiados	380
Respostas aos Exercícios Exploratórios	381
103.7 Lição 2	382
Introdução	382
O localizador de padrões: grep	382
O editor de fluxo: sed	386
Combinando grep e sed	390
Exercícios Guiados	393
Exercícios Exploratórios	394
Resumo	396
Respostas aos Exercícios Guiados	397
Respostas aos Exercícios Exploratórios	398

103.8 Edição básica de arquivos com o vi	400
103.8 Lição 1	401
Introdução	401
Modo de inserção	402
Modo normal	402
Comandos de dois pontos	405
Editores alternativos	406
Exercícios Guiados	408
Exercícios Exploratórios	409
Resumo	410
Respostas aos Exercícios Guiados	411
Respostas aos Exercícios Exploratórios	412
TÓPICO 104: DISPOSITIVOS, SISTEMAS DE ARQUIVOS LINUX E PADRÃO FHS	413
104.1 Criar partições e sistemas de arquivos	414
104.1 Lição 1	415
Introdução	415
Entendendo MBR e GPT	416
Criando sistemas de arquivos	423
Gerenciando Partições com o GNU Parted	434
Criando partições de troca	441
Exercícios Guiados	444
Exercícios Exploratórios	445
Resumo	447
Respostas aos Exercícios Guiados	448
Respostas aos Exercícios Exploratórios	449
104.2 Manutenção da integridade de sistemas de arquivos	451
104.2 Lição 1	452
Introdução	452
Verificando o uso de disco	453
Em busca de espaço livre	455
Manutenção de sistemas de arquivos ext2, ext3 e ext4	459
Exercícios Guiados	467
Exercícios Exploratórios	468
Resumo	469
Respostas aos Exercícios Guiados	470
Respostas aos Exercícios Exploratórios	472
104.3 Controle da montagem e desmontagem dos sistemas de arquivos	474
104.3 Lição 1	475
Introdução	475
Montando e desmontando sistemas de arquivos	475

Montagem de sistemas de arquivos na inicialização	479
Usando UUIDs e rótulos	482
Montando discos com Systemd	483
Exercícios Guiados	487
Exercícios Exploratórios	488
Resumo	489
Respostas aos Exercícios Guiados	490
Respostas aos Exercícios Exploratórios	492
104.5 Controlar permissões e propriedades de arquivos	494
104.5 Lição 1	495
Introdução	495
Consulta de informações sobre arquivos e diretórios	495
E quanto aos diretórios?	497
Exibindo arquivos ocultos	497
Entendendo os tipos de arquivos	498
Entendendo as permissões	499
Modificando as permissões de arquivos	501
Modificando o proprietário de um arquivo	504
Consultando os grupos	505
Permissões padrão	506
Permissões especiais	508
Exercícios Guiados	512
Exercícios Exploratórios	514
Resumo	515
Respostas aos Exercícios Guiados	516
Respostas aos Exercícios Exploratórios	519
104.6 Criar e alterar links simbólicos e hardlinks	522
104.6 Lição 1	523
Introdução	523
Compreendendo os links	523
Exercícios Guiados	528
Exercícios Exploratórios	529
Resumo	532
Respostas aos Exercícios Guiados	533
Respostas aos Exercícios Exploratórios	534
104.7 Encontrar arquivos de sistema e conhecer sua localização correta	538
104.7 Lição 1	539
Introdução	539
O Filesystem Hierarchy Standard	539
Busca de arquivos	542

Exercícios Guiados.....	551
Exercícios Exploratórios	552
Resumo	553
Respostas aos Exercícios Guiados.....	554
Respostas aos Exercícios Exploratórios	556
Imprint.....	558



Tópico 101: Arquitetura de Sistema



101.1 Identificar e editar configurações de hardware

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 101.1

Peso

2

Áreas chave de conhecimento

- Habilitar e desabilitar periféricos integrados.
- Diferenciar entre vários tipos de dispositivos de armazenamento.
- Determinar os recursos de hardware para os dispositivos.
- Ferramentas e utilitários para a listar várias informações de hardware (por exemplo, lsusb, lspci, etc...).
- Ferramentas e utilitários para manipular dispositivos USB.
- Compreensão conceitual de sysfs, udev e dbus.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- /sys/
- /proc/
- /dev/
- modprobe
- lsmod
- lspci
- lsusb



**Linux
Professional
Institute**

101.1 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	101 Arquitetura do sistema
Objetivo:	101.1 Determinar e definir configurações de hardware
Lição:	1 de 1

Introdução

Desde os primeiros anos da computação eletrônica, os fabricantes de computadores pessoais e empresariais vêm integrando uma série de peças de hardware em suas máquinas. Essas peças, por sua vez, devem ser suportadas pelo sistema operacional. Por isso, é preciso que existam padrões estabelecidos para os conjuntos de instruções e a comunicação dos dispositivos, semelhante à camada de abstração padronizada fornecida pelo sistema operacional a um aplicativo. Esses padrões facilitam a criação e a manutenção de um sistema operacional não vinculado a um modelo de hardware específico. No entanto, a complexidade do hardware subjacente integrado às vezes requer ajustes na maneira como os recursos devem ser expostos ao sistema operacional, para que ele possa ser instalado e funcionar corretamente.

É possível realizar alguns desses ajustes mesmo sem um sistema operacional instalado. A maioria das máquinas oferece um utilitário de configuração que pode ser executado quando a máquina é ligada. Até meados dos anos 2000, o utilitário de configuração era implementado na BIOS (*Basic Input/Output System*), o padrão de firmware contendo as rotinas básicas de configuração para as placas-mãe x86. A partir do final da primeira década dos anos 2000, as máquinas baseadas na arquitetura x86 começaram a substituir a BIOS por uma nova implementação chamada UEFI (*Unified*

Extensible Firmware Interface), que tem recursos mais sofisticados para identificação, teste, configuração e atualização de firmware. Apesar da mudança, não é incomum que se chame o utilitário de configuração de BIOS, já que ambas as implementações cumprem o mesmo objetivo básico.

NOTE

Trataremos em mais detalhes das diferenças entre a BIOS e a UEFI em uma lição posterior.

Ativação do dispositivo

O utilitário de configuração do sistema aparece quando pressionamos uma tecla específica ao ligar o computador. Essa tecla varia de fabricante para fabricante, mas geralmente é **Del** ou uma das teclas de função, como **F2** ou **F12**. A combinação de teclas a ser usada é frequentemente exibida na tela inicial.

No utilitário de configuração da BIOS, é possível ativar e desativar os periféricos integrados, ativar a proteção básica contra erros e alterar configurações de hardware como o IRQ (solicitação de interrupção) e o DMA (acesso direto à memória). Nas máquinas modernas, é raro que seja necessário alterar essas configurações, mas às vezes é preciso fazer ajustes para solucionar problemas específicos. Existem tecnologias de RAM, por exemplo, que são compatíveis com taxas de transferência de dados mais rápidas do que os valores padrão, e portanto é recomendável alterá-los para os valores especificados pelo fabricante. Algumas CPUs oferecem recursos que às vezes não são necessários para aquela instalação específica, podendo assim ser desativados. Isso reduz o consumo de energia e contribui para aumentar a proteção do sistema, já que os recursos da CPU que contenham bugs conhecidos também podem ser desativados.

Se a máquina estiver equipada com vários dispositivos de armazenamento, é importante definir qual deles possui o carregador de inicialização correto e deve ser o primeiro na ordem de inicialização do dispositivo. O sistema operacional pode não carregar se o dispositivo incorreto aparecer primeiro na lista de verificações de inicialização da BIOS.

Inspeção de dispositivos no Linux

Uma vez que os dispositivos são corretamente identificados, cabe ao sistema operacional associar os componentes de software correspondentes exigidos por eles. Quando um recurso de hardware não está funcionando como esperado, é importante identificar onde exatamente está o problema. Quando uma peça de hardware não é detectada pelo sistema operacional, é mais provável que o defeito esteja na peça ou na porta à qual está conectada. Quando a peça é detectada, mas não funciona corretamente, pode haver um problema no lado do sistema operacional. Portanto, uma das primeiras etapas ao lidar com problemas relacionados a hardware é verificar se o sistema operacional está detectando corretamente o dispositivo. Existem duas maneiras básicas de identificar recursos de

hardware em um sistema Linux: usar comandos especializados ou consultar arquivos específicos em sistemas de arquivos especiais.

Comandos para inspeção

Os dois comandos essenciais para identificar dispositivos conectados em um sistema Linux são:

lspci

Mostra todos os dispositivos atualmente conectados ao barramento PCI (*Peripheral Component Interconnect*). Os dispositivos PCI podem ser um componente conectado à placa-mãe, como um controlador de disco, ou uma placa de expansão instalada em um slot PCI, como uma placa de vídeo externa.

lsusb

Lista os dispositivos USB (*Universal Serial Bus*) atualmente conectados à máquina. Embora existam dispositivos USB para praticamente qualquer finalidade imaginável, a interface USB é amplamente usada para conectar dispositivos de entrada—teclados, dispositivos apontadores—e mídias de armazenamento removíveis.

A saída dos comandos `lspci` e `lsusb` consiste em uma lista de todos os dispositivos PCI e USB identificados pelo sistema operacional. No entanto, o dispositivo pode não estar totalmente operacional ainda, porque cada peça de hardware requer um componente de software para controlar o dispositivo correspondente. Esse componente de software é chamado de *módulo do kernel* e pode fazer parte do kernel oficial do Linux ou ser adicionado separadamente. Os módulos do kernel do Linux relacionados a dispositivos de hardware também são chamados de *drivers*, como em outros sistemas operacionais. Os drivers para Linux, no entanto, nem sempre são fornecidos pelo fabricante do dispositivo. Enquanto alguns fabricantes fornecem seus próprios drivers binários para serem instalados separadamente, muitos drivers são criados por desenvolvedores independentes. Historicamente, dispositivos que funcionam no Windows, por exemplo, podem não ter um módulo de kernel equivalente para Linux. Atualmente, os sistemas operacionais baseados em Linux têm um forte suporte de hardware e a maioria dos dispositivos funciona sem dar trabalho.

Os comandos diretamente relacionados ao hardware geralmente requerem privilégios de root para serem executados, exibindo apenas informações limitadas quando executados por um usuário normal; portanto, pode ser necessário fazer login como root ou executar o comando com `sudo`. A seguinte saída do comando `lspci`, por exemplo, mostra alguns dispositivos identificados:

```
§ lspci
01:00.0 VGA compatible controller: NVIDIA Corporation GM107 [GeForce GTX 750 Ti]
(rev a2)
04:02.0 Network controller: Ralink corp. RT2561/RT61 802.11g PCI
```

```
04:04.0 Multimedia audio controller: VIA Technologies Inc. ICE1712 [Envy24] PCI
Multi-Channel I/O Controller (rev 02)
04:0b.0 FireWire (IEEE 1394): LSI Corporation FW322/323 [TrueFire] 1394a Controller
(rev 70)
```

A saída desses comandos pode ter dezenas de linhas, de forma que o exemplo anterior e o seguinte incluem apenas as partes que nos interessam. Os números hexadecimais no início de cada linha são o endereço exclusivo do dispositivo PCI correspondente. O comando `lspci` mostra mais detalhes sobre um dispositivo específico se seu endereço for fornecido com a opção `-s`, acompanhada da opção `-v`:

```
$ lspci -s 04:02.0 -v
04:02.0 Network controller: Ralink corp. RT2561/RT61 802.11g PCI
    Subsystem: Linksys WMP54G v4.1
    Flags: bus master, slow devsel, latency 32, IRQ 21
    Memory at e3100000 (32-bit, non-prefetchable) [size=32K]
    Capabilities: [40] Power Management version 2
    kernel driver in use: rt61pci
```

A saída agora mostra muito mais detalhes sobre o dispositivo no endereço `04:02.0`. Trata-se de um controlador de rede cujo nome interno é `Ralink corp. RT2561/RT61 802.11g PCI`. `Subsystem` está associado à marca e modelo do dispositivo—`Linksys WMP54G v4.1`—e pode ser útil para fins de diagnóstico.

O módulo do kernel pode ser identificado na linha `kernel driver in use`, que mostra o módulo `rt61pci`. De todas as informações coletadas, é correto supor que:

1. O dispositivo foi identificado.
2. Um módulo do kernel correspondente foi carregado.
3. O dispositivo deve estar pronto para uso.

Outra maneira de verificar qual módulo do kernel está sendo usado para o dispositivo especificado seria usar a opção `-k`, disponível nas versões mais recentes do `lspci`:

```
$ lspci -s 01:00.0 -k
01:00.0 VGA compatible controller: NVIDIA Corporation GM107 [GeForce GTX 750 Ti]
(rev a2)
    kernel driver in use: nvidia
    kernel modules: nouveau, nvidia_drm, nvidia
```

Para o dispositivo escolhido, uma placa NVIDIA GPU, `lspci` informa que o módulo em uso é chamado `nvidia`, na linha `kernel driver in use: nvidia`, e todos os módulos correspondentes do kernel estão listados na linha `kernel modules: nouveau, nvidia_drm, nvidia`. O comando `lsusb` é semelhante a `lspci`, mas lista exclusivamente as informações relativas à USB:

```
$ lsusb
Bus 001 Device 029: ID 1781:0c9f Multiple Vendors USbtiny
Bus 001 Device 028: ID 093a:2521 Pixart Imaging, Inc. Optical Mouse
Bus 001 Device 020: ID 1131:1001 Integrated System Solution Corp. KY-BT100
Bluetooth Adapter
Bus 001 Device 011: ID 04f2:0402 Chicony Electronics Co., Ltd Genius LuxeMate i200
Keyboard
Bus 001 Device 007: ID 0424:7800 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 002: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

O comando `lsusb` mostra os canais USB disponíveis e os dispositivos conectados a eles. Como no caso do `lspci`, a opção `-v` exibe uma saída mais detalhada. Um dispositivo específico pode ser selecionado para inspeção, bastando fornecer seu ID com a opção `-d`:

```
$ lsusb -v -d 1781:0c9f
Bus 001 Device 029: ID 1781:0c9f Multiple Vendors USbtiny
Device Descriptor:
  bLength          18
  bDescriptorType   1
  bcdUSB         1.01
  bDeviceClass      255 Vendor Specific Class
  bDeviceSubClass     0
  bDeviceProtocol     0
  bMaxPacketSize0       8
  idVendor           0x1781 Multiple Vendors
  idProduct           0x0c9f USbtiny
  bcdDevice        1.04
  iManufacturer        0
  iProduct             2 USbtiny
  iSerial              0
  bNumConfigurations    1
```

Com a opção `-t`, o comando `lsusb` mostra os mapeamentos do dispositivo USB atual na forma de árvore hierárquica:

```
$ lsusb -t
/: Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/1p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M
        |__ Port 1: Dev 3, If 0, Class=Hub, Driver=hub/3p, 480M
            |__ Port 2: Dev 11, If 1, Class=Human Interface Device, Driver=usbhid,
1.5M
                |__ Port 2: Dev 11, If 0, Class=Human Interface Device, Driver=usbhid,
1.5M
                    |__ Port 3: Dev 20, If 0, Class=Wireless, Driver=btusb, 12M
                    |__ Port 3: Dev 20, If 1, Class=Wireless, Driver=btusb, 12M
                    |__ Port 3: Dev 20, If 2, Class=Application Specific Interface,
Driver=, 12M
                        |__ Port 1: Dev 7, If 0, Class=Vendor Specific Class, Driver=lan78xx,
480M
                            |__ Port 2: Dev 28, If 0, Class=Human Interface Device, Driver=usbhid, 1.5M
                            |__ Port 3: Dev 29, If 0, Class=Vendor Specific Class, Driver=, 1.5M
```

É possível que nem todos os dispositivos tenham um módulo correspondente associado. A comunicação com determinados dispositivos pode ser feita diretamente pelo aplicativo, sem a intermediação de um módulo. Ainda assim, existem informações importantes na saída de `lsusb -t`. Quando existe um módulo correspondente, seu nome aparece no final da linha do dispositivo, como em `Driver=btusb`. O dispositivo `Class` identifica a categoria geral, como `Human Interface Device`, `Wireless`, `Vendor Specific Class`, `Mass Storage`, dentre outros. Para verificar qual dispositivo está usando o módulo `btusb`, presente na lista anterior, os números de `Bus` e `Dev` devem ser fornecidos à opção `-s` do comando `lsusb`:

```
$ lsusb -s 01:20
Bus 001 Device 020: ID 1131:1001 Integrated System Solution Corp. KY-BT100
Bluetooth Adapter
```

É comum que haja um grande conjunto de módulos de kernel carregados em um sistema Linux padrão a qualquer momento. Para interagir com eles, o melhor jeito é usar os comandos fornecidos pelo pacote `kmod`, um conjunto de ferramentas para realizar tarefas comuns com os módulos do kernel Linux, como inserir, remover, listar, verificar propriedades, resolver dependências e aliases. O comando `lsmod`, por exemplo, mostra todos os módulos carregados no momento:

```
$ lsmod
Module           Size  Used by
kvm_intel       138528  0
kvm             421021  1 kvm_intel
iTCO_wdt        13480   0
```

iTCO_vendor_support	13419	1	iTCO_wdt
snd_usb_audio	149112	2	
snd_hda_codec_realtek	51465	1	
snd_ice1712	75006	3	
snd_hda_intel	44075	7	
arc4	12608	2	
snd_cs8427	13978	1	snd_ice1712
snd_i2c	13828	2	snd_ice1712,snd_cs8427
snd_ice17xx_ak4xxx	13128	1	snd_ice1712
snd_ak4xxx_adda	18487	2	snd_ice1712,snd_ice17xx_ak4xxx
microcode	23527	0	
snd_usbmidi_lib	24845	1	snd_usb_audio
gspca_pac7302	17481	0	
gspca_main	36226	1	gspca_pac7302
videodev	132348	2	gspca_main,gspca_pac7302
rt61pci	32326	0	
rt2x00pci	13083	1	rt61pci
media	20840	1	videodev
rt2x00mmio	13322	1	rt61pci
hid_dr	12776	0	
snd_mpu401_uart	13992	1	snd_ice1712
rt2x00lib	67108	3	rt61pci,rt2x00pci,rt2x00mmio
snd_rawmidi	29394	2	snd_usbmidi_lib,snd_mpu401_uart

A saída do comando `lsmod` é dividida em três colunas:

Module

Nome do módulo.

Size

Quantidade de RAM utilizada pelo módulo, em bytes.

Used by

Módulos dependentes.

Alguns módulos exigem que outros módulos funcionem corretamente, como é o caso dos módulos para dispositivos de áudio:

```
$ lsmod | grep -i snd_hda_intel
snd_hda_intel          42658  5
snd_hda_codec          155748  3  snd_hda_codec_hdmi,snd_hda_codec_via,snd_hda_intel
snd_pcm                81999  3  snd_hda_codec_hdmi,snd_hda_codec,snd_hda_intel
```

```

snd_page_alloc      13852  2 snd_pcm,snd_hda_intel
snd                59132  19
snd_hwdep,snd_timer,snd_hda_codec_hdmi,snd_hda_codec_via,snd_pcm,snd_seq,snd_hda_codec,snd_hda_intel,snd_seq_device

```

A terceira coluna, `Used by`, mostra os módulos que exigem que o módulo na primeira coluna funcione corretamente. Muitos módulos da arquitetura de som do Linux, prefixados por `snd`, são interdependentes. Ao procurar por problemas durante um diagnóstico do sistema, pode ser útil descarregar módulos específicos atualmente carregados. O comando `modprobe` pode ser usado para carregar e descarregar módulos do kernel: para descarregar um módulo e seus módulos relacionados, desde que não estejam sendo usados por um processo em execução, use o comando `modprobe -r`. Por exemplo, para descarregar o módulo `snd-hda-intel` (o módulo para um dispositivo de áudio HDA Intel) e outros módulos relacionados ao sistema de som:

```
# modprobe -r snd-hda-intel
```

Além de carregar e descarregar módulos do kernel enquanto o sistema está em execução, é possível alterar os parâmetros do módulo quando o kernel está sendo carregado, o que não é muito diferente de passar opções para os comandos. Os módulos aceitam parâmetros específicos, mas na maioria das vezes recomenda-se usar os valores padrão, não sendo necessários parâmetros extras. No entanto, em alguns casos precisamos usar parâmetros para alterar o comportamento de um módulo e fazê-lo funcionar conforme o esperado.

Usando o nome do módulo como único argumento, o comando `modinfo` mostra uma descrição, o arquivo, o autor, a licença, a identificação, as dependências e os parâmetros disponíveis para o módulo fornecido. Para que os parâmetros personalizados de um módulo se tornem persistentes, inclua-os no arquivo `/etc/modprobe.conf` ou em arquivos individuais com a extensão `.conf` no diretório `/etc/modprobe.d/``. A opção `-p` fará com que o comando `modinfo` exiba todos os parâmetros disponíveis e ignore as outras informações:

```

# modinfo -p nouveau
vram_pushbuf:Create DMA push buffers in VRAM (int)
tv_norm:Default TV norm.
          Supported: PAL, PAL-M, PAL-N, PAL-Nc, NTSC-M, NTSC-J,
                     hd480i, hd480p, hd576i, hd576p, hd720p, hd1080i.
          Default: PAL
          NOTE Ignored for cards with external TV encoders. (charp)
nofbaccel:Disable fbcon acceleration (int)
fbcon_bpp:fbcon bits-per-pixel (default: auto) (int)
mst:Enable DisplayPort multi-stream (default: enabled) (int)
tv_disable:Disable TV-out detection (int)

```

```

ignorelid:Ignore ACPI lid status (int)
duallink:Allow dual-link TMDS (default: enabled) (int)
hdmi_mhz:Force a maximum HDMI pixel clock (in MHz) (int)
config:option string to pass to driver core (charp)
debug:debug string to pass to driver core (charp)
noaccel:disable kernel/abi16 acceleration (int)
modeset:enable driver (default: auto, 0 = disabled, 1 = enabled, 2 = headless) (int)
atomic:Expose atomic ioctl (default: disabled) (int)
runpm:disable (0), force enable (1), optimus only default (-1) (int)

```

O exemplo de saída mostra todos os parâmetros disponíveis para o módulo `nouveau`, um módulo do kernel fornecido pelo *nouveau project* como alternativa aos drivers proprietários das placas de GPU da NVIDIA. A opção `modeset`, por exemplo, permite controlar se a resolução e a profundidade da tela serão definidas no espaço do kernel em vez do espaço do usuário. Quando adicionamos `options nouveau modeset =0` ao arquivo `/etc/modprobe.d/nouveau.conf`, o recurso `modeset` do kernel é desativado.

Se um módulo estiver causando problemas, o arquivo `/etc/modprobe.d/blacklist.conf` pode ser usado. Por exemplo, para impedir o carregamento automático do módulo `nouveau`, a linha `blacklist nouveau` deve ser adicionada ao arquivo `/etc/modprobe.d/blacklist.conf`. Essa ação é necessária quando o módulo proprietário `nvidia` é instalado e o módulo padrão `nouveau` deve ser posto de lado.

NOTE

É possível modificar o arquivo `/etc/modprobe.d/blacklist.conf` que já existe no sistema por padrão. Porém, a melhor opção é criar um arquivo de configuração separado, `/etc/modprobe.d/<module_name>.conf`, contendo ajustes específicos àquele módulo do kernel em particular.

Arquivos de informação e de dispositivo

Os comandos `lspci`, `lsusb` e `lsmod` atuam como front-ends para ler as informações de hardware armazenadas pelo sistema operacional. Este tipo de informação é mantido em arquivos especiais nos diretórios `/proc` e `/sys`. Esses diretórios são pontos de montagem para sistemas de arquivos que não estão presentes em uma partição de dispositivo, mas somente no espaço de RAM usado pelo kernel para armazenar a configuração do tempo de execução e informações sobre os processos em execução. Esses sistemas de arquivos não se destinam ao armazenamento convencional de arquivos e, portanto, são chamados de pseudosistemas de arquivos e existem apenas enquanto o sistema estiver em execução. O diretório `/proc` contém arquivos com informações sobre processos em execução e recursos de hardware. Alguns dos arquivos importantes em `/proc` para a inspeção de hardware são:

/proc/cpuinfo

Lista informações detalhadas sobre a(s) CPU(s) encontradas pelo sistema operacional.

/proc/interrupts

Uma lista de números de interrupções por dispositivo de entrada e saída em cada CPU.

/proc/ioports

Lista as regiões de portas de Entrada/Saída registradas atualmente e em uso.

/proc/dma

Lista os canais registrados de DMA (acesso direto à memória) em uso.

Os arquivos dentro do diretório `/sys` têm funções semelhantes às do `/proc`. No entanto, o diretório `/sys` tem o objetivo específico de armazenar informações do dispositivo e dados do kernel relacionados ao hardware, ao passo que `/proc` também contém informações sobre diversas estruturas de dados do kernel, incluindo processos em execução e configurações.

Outro diretório diretamente relacionado aos dispositivos em um sistema Linux padrão é o `/dev`. Cada arquivo dentro de `/dev` é associado a um dispositivo do sistema, particularmente dispositivos de armazenamento. Um disco rígido IDE legado, por exemplo, quando conectado ao primeiro canal IDE da placa-mãe, é representado pelo arquivo `/dev/hda`. Cada partição desse disco será identificada por `/dev/hda1`, `/dev/hda2` e assim por diante, até a última partição encontrada.

Os dispositivos removíveis são manipulados pelo subsistema `udev`, que cria os dispositivos correspondentes em `/dev`. O kernel do Linux captura o evento de detecção de hardware e o passa para o processo `udev`, que por sua vez identifica o dispositivo e cria dinamicamente os arquivos correspondentes em `/dev`, usando regras predefinidas.

Nas distribuições Linux atuais, o `udev` é responsável pela identificação e configuração dos dispositivos já presentes durante a inicialização da máquina (*detecção coldplug*) e dos dispositivos identificados enquanto o sistema está em execução (*detecção hotplug*). O `Udev` utiliza o `SysFS`, o pseudosistema de arquivos montado em `/sys` para informações relacionadas ao hardware.

NOTE

Hotplug é o termo usado para se referir à detecção e configuração de um dispositivo enquanto o sistema está em execução, como quando inserimos um dispositivo USB. O kernel do Linux suporta recursos de hotplug desde a versão 2.6, permitindo que a maioria dos barramentos do sistema (PCI, USB, etc.) disparem eventos de hotplug quando um dispositivo é conectado ou desconectado.

À medida que novos dispositivos são detectados, o `udev` pesquisa uma regra correspondente nas regras predefinidas armazenadas no diretório `/etc/udev/rules.d/`. As regras mais importantes são

fornecidas pela distribuição, mas é possível adicionar novas para casos específicos.

Dispositivos de armazenamento

No Linux, os dispositivos de armazenamento são genericamente chamados de dispositivos de bloco, porque os dados que contêm são lidos em blocos de dados armazenados em buffer com diferentes tamanhos e posições. Cada dispositivo de bloco é identificado por um arquivo no diretório `/dev`, sendo que o nome do arquivo depende do tipo de dispositivo (IDE, SATA, SCSI, etc.) e de suas partições. Os dispositivos de CD/DVD e de disquete, por exemplo, receberão nomes específicos em `/dev`: uma unidade de CD/DVD conectada ao segundo canal IDE será identificada como `/dev/hdc` (`/dev/hda` e `/dev/hdb` são reservados para os dispositivos mestre e escravo no primeiro canal IDE) e uma unidade de disquete antiga será identificada como `/dev/fd0`, `/dev/fd1`, etc.

A partir da versão 2.4 do kernel Linux em diante, a maioria dos dispositivos de armazenamento passou a ser identificada como dispositivos SCSI, independentemente do tipo de hardware. Os dispositivos de bloco IDE, SSD e USB são prefixados com `sd`. Para os discos IDE, o prefixo `sd` é usado, mas a terceira letra é escolhida dependendo da unidade ser mestre ou escrava (no primeiro canal IDE, o mestre será `sda` e o escravo será `sdb`). As partições são listadas em ordem numérica. Os caminhos `/dev/sda1`, `/dev/sda2`, etc. são usados para a primeira e a segunda partições do dispositivo de bloco identificado primeiro e `/dev/sdb1`, `/dev/sdb2`, etc. identificam a primeira e a segunda partições do dispositivo de bloco identificado a seguir. A exceção a esse padrão ocorre com cartões de memória (cartões SD) e dispositivos NVMe (SSD conectados ao barramento PCI Express). Para os cartões SD, os caminhos `/dev/mmcblk0p1`, `/dev/mmcblk0p2` etc. são usados para a primeira e a segunda partições do dispositivo identificado primeiro e `/dev/mmcblk1p1`, `/dev/mmcblk1p2`, etc. identificam a primeira e a segunda partições do dispositivo identificado em segundo lugar. Os dispositivos NVMe recebem o prefixo `nvme`, como em `/dev/nvme0n1p1` e `/dev/nvme0n1p2`.

Exercícios Guiados

1. Suponha que um sistema operacional não consegue inicializar após um segundo disco SATA ser adicionado ao sistema. Sabendo que as peças não são defeituosas, qual poderia ser a causa possível desse erro?

2. Você acaba de adquirir um novo computador de mesa e gostaria de conferir se a placa de vídeo externa conectada ao barramento PCI é realmente a anunciada pelo fabricante. Porém, se abrir o gabinete do computador, a garantia será anulada. Qual comando pode ser usado para listar as informações da placa de vídeo detectadas pelo sistema operacional?

3. A linha a seguir faz parte da saída gerada pelo comando `lspci`:

```
03:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 2208  
[Thunderbolt] (rev 05)
```

Qual comando deve ser executado para identificar o módulo do kernel em uso neste dispositivo específico?

4. Um administrador deseja experimentar parâmetros diferentes para o módulo do kernel `bluetooth` sem reiniciar o sistema. No entanto, qualquer tentativa de descarregar o módulo com `modprobe -r bluetooth` resulta no seguinte erro:

```
modprobe: FATAL: Module bluetooth is in use.
```

Qual a possível causa desse erro?

Exercícios Exploratórios

- É comum encontrar máquinas legadas em ambientes de produção, por exemplo quando certos equipamentos usam uma conexão desatualizada para se comunicar com o computador controlador, sendo assim necessário estar particularmente atento a algumas peculiaridades dessas máquinas mais antigas. Certos servidores x86 com firmware BIOS mais antigo, por exemplo, não inicializam se um teclado não for detectado. Como esse problema específico pode ser evitado?

- Os sistemas operacionais criados em torno do kernel do Linux também estão disponíveis para uma ampla variedade de arquiteturas de computadores além do x86, como os computadores de placa única baseados na arquitetura ARM. Um usuário atento notará a ausência do comando `lspci` nessas máquinas, como o Raspberry Pi. Que diferença em relação às máquinas x86 justifica essa ausência?

- Muitos roteadores de rede incluem uma porta USB que permite a conexão de um dispositivo externo, como um disco rígido USB. Sabendo que a maioria deles usa um sistema operacional baseado em Linux, qual seria o nome de um disco rígido USB externo no diretório `/dev/`, supondo-se que não haja nenhum outro dispositivo de bloco convencional no roteador?

- Em 2018, a vulnerabilidade de hardware conhecida como *Meltdown* foi descoberta. Ela afeta quase todos os processadores de diferentes arquiteturas. As versões mais recentes do kernel Linux podem informar se o sistema atual está vulnerável. Como obter essas informações?

Resumo

Esta lição aborda os conceitos gerais de como o kernel do Linux lida com recursos de hardware, principalmente na arquitetura x86. A lição inclui os seguintes tópicos:

- Como as configurações definidas nos utilitários de configuração da BIOS ou UEFI podem afetar a interação do sistema operacional com o hardware.
- Como usar as ferramentas fornecidas por um sistema Linux padrão para obter informações sobre o hardware.
- Como identificar dispositivos de armazenamento permanentes e removíveis no sistema de arquivos. Os comandos e procedimentos abordados foram:
- Comandos para inspecionar o hardware detectado: `lspci` and `lsusb`.
- Comandos para gerenciar os módulos do kernel: `lsmod` and `modprobe`.
- Arquivos especiais relacionados ao hardware, seja os arquivos encontrados no diretório `/dev/` ou nos pseudosistemas de arquivos `/proc/` e `/sys/`.

Respostas aos Exercícios Guiados

- Suponha que um sistema operacional não consegue inicializar após um segundo disco SATA ser adicionado ao sistema. Sabendo que as peças não são defeituosas, qual poderia ser a causa possível desse erro?

A ordem dos dispositivos de inicialização deve ser definida no utilitário de configuração da BIOS, caso contrário a BIOS pode não conseguir executar o carregador de inicialização.

- Você acaba de adquirir um novo computador de mesa e gostaria de conferir se a placa de vídeo externa conectada ao barramento PCI é realmente a anunciada pelo fabricante. Porém, se abrir o gabinete do computador, a garantia será anulada. Qual comando pode ser usado para listar as informações da placa de vídeo detectadas pelo sistema operacional?

O comando `lspci` lista informações detalhadas sobre todos os dispositivos atualmente conectados ao barramento PCI.

- A linha a seguir faz parte da saída gerada pelo comando `lspci`:

```
03:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 2208
[Thunderbolt] (rev 05)
```

Qual comando deve ser executado para identificar o módulo do kernel em uso neste dispositivo específico?

O comando `lspci -s 03:00.0 -v` ou `lspci -s 03:00.0 -k`

- Um administrador deseja experimentar parâmetros diferentes para o módulo do kernel `bluetooth` sem reiniciar o sistema. No entanto, qualquer tentativa de descarregar o módulo com `modprobe -r bluetooth` resulta no seguinte erro:

```
modprobe: FATAL: Module bluetooth is in use.
```

Qual a possível causa desse erro?

O módulo `bluetooth` está sendo usado por um processo em execução.

Respostas aos Exercícios Exploratórios

- É comum encontrar máquinas legadas em ambientes de produção, por exemplo quando certos equipamentos usam uma conexão desatualizada para se comunicar com o computador controlador, sendo assim necessário estar particularmente atento a algumas peculiaridades dessas máquinas mais antigas. Certos servidores x86 com firmware BIOS mais antigo, por exemplo, não inicializam se um teclado não for detectado. Como esse problema específico pode ser evitado?

O utilitário de configuração da BIOS tem uma opção para desativar o bloqueio do computador quando um teclado não é encontrado.

- Os sistemas operacionais criados em torno do kernel do Linux também estão disponíveis para uma ampla variedade de arquiteturas de computadores além do x86, como os computadores de placa única baseados na arquitetura ARM. Um usuário atento notará a ausência do comando `lspci` nessas máquinas, como o Raspberry Pi. Que diferença em relação às máquinas x86 justifica essa ausência?

Ao contrário da maioria das máquinas x86, um computador baseado em ARM, como o Raspberry Pi, não possui um barramento PCI; portanto, o comando `lspci` é inútil.

- Muitos roteadores de rede incluem uma porta USB que permite a conexão de um dispositivo externo, como um disco rígido USB. Sabendo que a maioria deles usa um sistema operacional baseado em Linux, qual seria o nome de um disco rígido USB externo no diretório `/dev/`, supondo-se que não haja nenhum outro dispositivo de bloco convencional no roteador?

Os kernels do Linux modernos identificam os discos rígidos USB como dispositivos SATA, de modo que o arquivo correspondente será `/dev/sda`, já que não existe nenhum outro dispositivo de bloco convencional no sistema.

- Em 2018, a vulnerabilidade de hardware conhecida como *Meltdown* foi descoberta. Ela afeta quase todos os processadores de diferentes arquiteturas. As versões mais recentes do kernel Linux podem informar se o sistema atual está vulnerável. Como obter essas informações?

O arquivo `/proc/cpuinfo` tem uma linha que mostra os bugs conhecidos para a CPU correspondente, como por exemplo bugs: `cpu_meltdown`.



101.2 Início (boot) do sistema

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 101.2

Peso

3

Áreas chave de conhecimento

- Fornecer os comandos e opções mais comuns para o gerenciador de inicialização e para o kernel durante a inicialização.
- Demonstrar conhecimento sobre a sequência de inicialização do BIOS/UEFI até sua conclusão.
- Entendimento do SysVinit e do systemd.
- Noções do Upstart.
- Conferir os arquivos de log dos eventos de inicialização.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- dmesg
- journalctl
- BIOS
- UEFI
- bootloader
- kernel
- initramfs

- `init`
- SysVinit
- systemd



101.2 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	101 Arquitetura do sistema
Objetivo:	101.2 Inicialização do sistema
Lição:	1 de 1

Introdução

Para controlar a máquina, o componente principal do sistema operacional – o kernel – deve ser carregado por um programa chamado *bootloader* (carregador de inicialização), que por sua vez é carregado por um firmware pré-instalado, como a BIOS ou a UEFI. O carregador de inicialização pode ser personalizado para passar parâmetros para o kernel, como a partição que contém o sistema de arquivos raiz ou em qual modo o sistema operacional deve ser executado. Uma vez carregado, o kernel dá seguimento ao processo de inicialização, identificando e configurando o hardware. Por fim, o kernel chama o utilitário responsável por iniciar e gerenciar os serviços do sistema.

NOTE

Em algumas distribuições Linux, os comandos executados nesta lição podem exigir privilégios de root.

BIOS ou UEFI

Os procedimentos executados pelas máquinas x86 para executar o carregador de inicialização são diferentes conforme se usa a BIOS ou a UEFI. A BIOS, abreviação de *Basic Input/Output System*, é um programa armazenado em um chip de memória não volátil conectado à placa-mãe, executado sempre que o computador é ligado. Esse tipo de programa é chamado *firmware* e seu local de armazenamento é separado dos outros dispositivos de armazenamento do sistema. A BIOS pressupõe que os primeiros 440 bytes no primeiro dispositivo de armazenamento – seguindo a ordem definida no utilitário de configuração da BIOS – são o primeiro estágio do bootloader (também chamado de *bootstrap*). Os primeiros 512 bytes de um dispositivo de armazenamento são o que se chama de MBR (*Master Boot Record*) dos dispositivos de armazenamento que usam o esquema de partição DOS padrão e, além do primeiro estágio do gerenciador de inicialização, eles contêm a tabela de partições. Se o MBR não contiver os dados corretos, o sistema não poderá inicializar, a menos que um método alternativo seja empregado.

De um modo geral, as etapas pré-operacionais para inicializar um sistema equipado com BIOS são:

1. O processo POST (*power-on self-test*) é executado para identificar falhas simples de hardware assim que a máquina é ligada.
2. A BIOS ativa os componentes básicos para carregar o sistema, como a saída de vídeo, o teclado e as mídias de armazenamento.
3. A BIOS carrega o primeiro estágio do bootloader a partir do MBR (os primeiros 440 bytes do primeiro dispositivo, conforme definido no utilitário de configuração da BIOS).
4. O primeiro estágio do bootloader chama o segundo estágio, responsável por apresentar as opções de inicialização e carregar o kernel.

A UEFI, abreviação de *Unified Extensible Firmware Interface*, difere da BIOS em alguns pontos-chave. Como a BIOS, a UEFI também é um firmware, mas pode identificar partições e ler muitos sistemas de arquivos nelas. A UEFI não depende do MBR, levando em consideração apenas as configurações armazenadas na memória não-volátil (*NVRAM*) conectada à placa-mãe. Essas definições indicam a localização dos programas compatíveis com a UEFI, chamados *aplicativos EFI*, que serão executados automaticamente ou chamados a partir de um menu de inicialização. Os aplicativos EFI podem ser carregadores de inicialização, seletores de sistema operacional, ferramentas para diagnóstico e reparo do sistema etc. Eles devem estar em uma partição de um dispositivo de armazenamento convencional e em um sistema de arquivos compatível. Os sistemas de arquivos compatíveis padrão são FAT12, FAT16 e FAT32 para dispositivos de bloco e ISO-9660 para mídia ótica. Essa abordagem permite a implementação de ferramentas muito mais sofisticadas do que as que seriam possíveis com a BIOS.

A partição que contém os aplicativos EFI é chamada de *Partição de Sistema EFI* ou apenas ESP. Essa partição não deve ser compartilhada com outros sistemas de arquivos do sistema, como o sistema de

arquivos raiz ou os sistemas de arquivos de dados do usuário. O diretório EFI na partição ESP contém os aplicativos apontados pelas entradas salvas na NVRAM.

De maneira geral, as etapas de inicialização do pré-sistema operacional em um sistema com UEFI são:

1. O processo POST (*power-on self-test*) é executado para identificar falhas simples de hardware assim que a máquina é ligada.
2. A UEFI ativa os componentes básicos para carregar o sistema, como a saída de vídeo, o teclado e as mídias de armazenamento.
3. O firmware da UEFI lê as definições armazenadas na NVRAM para executar o aplicativo EFI predefinido armazenado no arquivo de sistemas da partição ESP. Normalmente, o aplicativo EFI predefinido é um bootloader.
4. Se o aplicativo EFI predefinido for um bootloader, ele carrega o kernel para iniciar o sistema operacional.

O padrão UEFI também suporta um recurso chamado *Inicialização Segura*, que permite apenas a execução de aplicativos EFI assinados, ou seja, aplicativos EFI autorizados pelo fabricante do hardware. Esse recurso aumenta a proteção contra software malicioso, mas pode dificultar a instalação de sistemas operacionais não cobertos pela garantia do fabricante.

O bootloader

O carregador de inicialização mais popular para Linux na arquitetura x86 é o GRUB (*Grand Unified Bootloader*). Assim que é chamado pela BIOS ou pela UEFI, o GRUB exibe uma lista de sistemas operacionais disponíveis para inicialização. Às vezes, a lista não aparece automaticamente, mas ela pode ser invocada pressionando `Shift` enquanto o GRUB está sendo chamado pela BIOS. Nos sistemas UEFI, a tecla a ser usada é `Esc`.

No menu do GRUB, é possível escolher qual dos kernels instalados deve ser carregado e passar novos parâmetros para ele. A maioria dos parâmetros do kernel segue o padrão `opção=valor`. Alguns dos parâmetros mais úteis do kernel são:

`acpi`

Ativa/desativa o suporte a ACPI. `acpi=off` desabilita o suporte a ACPI.

`init`

Define um iniciador de sistema alternativo. Por exemplo, `init=/bin/bash` define o shell Bash como iniciador. Assim, uma sessão do shell será iniciada logo após o processo de inicialização do

kernel.

systemd.unit

Define o destino do *systemd* a ser ativado. Por exemplo, `systemd.unit=graphical.target`. O *systemd* também aceita os níveis de execução numéricos definidos para *SysV*. Para ativar o nível de execução 1, por exemplo, é necessário apenas incluir o número 1 ou a letra S (abreviação de “single”) como parâmetro do kernel.

mem

Define a quantidade de RAM disponível para o sistema. Este parâmetro é útil para limitar a RAM disponível para cada convidado em uma máquina virtual. Assim, `mem=512M` limita a 512 megabytes a quantidade de RAM disponível para um sistema convidado em particular.

maxcpus

Limita o número de processadores (ou núcleos de processador) visíveis ao sistema em máquinas multiprocessador simétricas. Também é útil para máquinas virtuais. Um valor de 0 desativa o suporte a máquinas multiprocessador e tem o mesmo efeito do parâmetro do kernel `nosmp`. O parâmetro `maxcpus=2` limita a dois o número de processadores disponíveis para o sistema operacional.

quiet

Oculta a maioria das mensagens de inicialização.

vga

Seleciona um modo de vídeo. O parâmetro `vga=ask` mostra uma lista dos modos disponíveis a escolher.

root

Define a partição raiz, diferente da que está configurada no bootloader. Por exemplo, `root=/dev/sda3`.

rootflags

Opções de montagem para o arquivo de sistemas raiz.

ro

Torna somente para leitura a montagem inicial do arquivo de sistemas raiz.

rw

Permite escrever no arquivo de sistemas raiz durante a montagem inicial.

Geralmente não é necessário alterar os parâmetros do kernel, mas isso pode ser útil para detectar e resolver problemas relacionados ao sistema operacional. Os parâmetros do kernel devem ser adicionados ao arquivo `/etc/default/grub` na linha `GRUB_CMDLINE_LINUX` para que persistam após a inicialização. É necessário gerar um novo arquivo de configuração para o carregador de inicialização a cada vez que `/etc/default/grub` é alterado, o que é feito com o comando `grub-mkconfig -o /boot/grub/grub.cfg`. Quando o sistema operacional estiver rodando, os parâmetros do kernel usados para carregar a sessão ficam disponíveis para leitura no arquivo `/proc/cmdline`.

NOTE A configuração do GRUB será discutida mais adiante, em outra lição.

Inicialização do sistema

Além do kernel, o sistema operacional depende de outros componentes que fornecem os recursos esperados. Muitos desses componentes são carregados durante o processo de inicialização do sistema e vão de simples scripts de shell a programas de serviços mais complexos. Os scripts geralmente são usados para executar tarefas de curta duração que serão finalizadas durante o processo de inicialização. Os serviços, também conhecidos como *daemons*, podem ficar ativos o tempo todo, pois muitas vezes são responsáveis por aspectos específicos do sistema operacional.

Existe uma enorme diversidade de maneiras de incorporar scripts de inicialização e daemons com as mais diferentes características em uma distribuição Linux. Esse fato, historicamente, impediu o desenvolvimento de uma solução única que atenda às expectativas dos mantenedores e usuários de todas as distribuições Linux. No entanto, qualquer ferramenta escolhida pelos mantenedores de distribuições para executar esta função será capaz de pelo menos iniciar, interromper e reiniciar os serviços do sistema. Essas ações geralmente são executadas pelo próprio sistema após uma atualização de software, por exemplo, mas o administrador do sistema quase sempre precisa reiniciar o serviço manualmente após fazer modificações em seu arquivo de configuração.

Também é conveniente que um administrador do sistema possa ativar um conjunto específico de daemons, dependendo das circunstâncias. Por exemplo, deve ser possível executar apenas um conjunto mínimo de serviços para executar tarefas de manutenção do sistema.

NOTE

Estritamente falando, o sistema operacional é apenas o kernel e seus componentes, que controlam o hardware e gerenciam todos os processos. É comum, no entanto, usar o termo “sistema operacional” de maneira mais vaga, para designar um grupo inteiro de programas distintos que compõem o ambiente de software onde o usuário pode executar as tarefas computacionais básicas.

A inicialização do sistema operacional começa quando o carregador de inicialização carrega o kernel na RAM. Nesse momento, o kernel assume o controle da CPU e começa a detectar e configurar os aspectos fundamentais do sistema operacional, como a configuração básica de hardware e o

endereçamento de memória.

O kernel abre então o *initramfs* (*initial RAM filesystem*). O initramfs é um arquivo que contém um sistema de arquivos raiz temporário usado durante o processo de inicialização. O principal objetivo de um arquivo initramfs é fornecer os módulos necessários para que o kernel possa acessar o sistema de arquivos raiz “de verdade” do sistema operacional.

Logo que o sistema de arquivos raiz fica disponível, o kernel monta todos os sistemas de arquivos configurados em `/etc/fstab` e, em seguida, executa o primeiro programa, um utilitário chamado `init`. O programa `init` é responsável por executar todos os scripts de inicialização e daemons do sistema. Existem implementações distintas desses iniciadores de sistema além do tradicional `init`, como o `systemd` e o `Upstart`. Depois que o programa `init` é carregado, o initramfs é removido da RAM.

Padrão SysV

Um gerenciador de serviços baseado no padrão SysVinit controla quais daemons e recursos estarão disponíveis empregando o conceito de *níveis de execução*. Os níveis de execução são numerados de 0 a 6 e são projetados pelos mantenedores da distribuição para atender a propósitos específicos. As únicas definições de nível de execução compartilhadas entre todas as distribuições são os níveis 0, 1 e 6.

systemd

O `systemd` é um gerenciador de sistemas e serviços moderno, com uma camada de compatibilidade para os comandos e níveis de execução do SysV. O `systemd` possui uma estrutura concorrente, emprega sockets e D-Bus para a ativação de serviços, execução de daemon sob demanda, monitoramento de processos com `cgroups`, snapshot support, recuperação da sessão do sistema, controle de ponto de montagem e um controle de serviços baseado em dependências. Nos últimos anos, a maioria das grandes distribuições Linux adotou gradualmente o `systemd` como seu gerenciador de sistema padrão.

Upstart

Como o `systemd`, o Upstart é um substituto para o `init`. O foco do Upstart é acelerar o processo de inicialização, paralelizando o processo de carregamento dos serviços do sistema. O Upstart foi usado pelas distribuições baseadas no Ubuntu em versões anteriores, mas hoje deu lugar ao `systemd`.

Inspeção da inicialização

Às vezes ocorrem erros durante o processo de inicialização, mas nem sempre eles são críticos a ponto de interromper completamente o sistema operacional. Não obstante, esses erros podem comprometer o comportamento esperado do sistema. Todos os erros resultam em mensagens que

podem ser usadas para investigações futuras, pois elas contêm informações valiosas sobre quando e como o erro ocorreu. Mesmo quando nenhuma mensagem de erro é gerada, as informações coletadas durante o processo de inicialização podem ser úteis para fins de ajuste e configuração.

O espaço de memória em que o kernel armazena suas mensagens, incluindo as mensagens de inicialização, é chamado de *buffer de anel do kernel*. As mensagens são mantidas nesse buffer mesmo quando não são exibidas durante o processo de inicialização (por exemplo, quando uma animação é exibida em vez da mensagem). No entanto, o buffer de anel do kernel perde todas as mensagens quando o sistema é desligado ou se o comando `dmesg --clear` for executado. Sem opções, o comando `dmesg` exibe as mensagens atuais no buffer de anel do kernel:

```
$ dmesg
[    5.262389] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts:
(null)
[    5.449712] ip_tables: (C) 2000-2006 Netfilter Core Team
[    5.460286] systemd[1]: systemd 237 running in system mode.
[    5.480138] systemd[1]: Detected architecture x86-64.
[    5.481767] systemd[1]: Set hostname to <torre>.
[    5.636607] systemd[1]: Reached target User and Group Name Lookups.
[    5.636866] systemd[1]: Created slice System Slice.
[    5.637000] systemd[1]: Listening on Journal Audit Socket.
[    5.637085] systemd[1]: Listening on Journal Socket.
[    5.637827] systemd[1]: Mounting POSIX Message Queue File System...
[    5.638639] systemd[1]: Started Read required files in advance.
[    5.641661] systemd[1]: Starting Load Kernel Modules...
[    5.661672] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
[    5.694322] lp: driver loaded but no devices found
[    5.702609] ppdev: user-space parallel port driver
[    5.705384] parport_pc 00:02: reported by Plug and Play ACPI
[    5.705468] parport0: PC-style at 0x378 (0x778), irq 7, dma 3
[PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
[    5.800146] lp0: using parport0 (interrupt-driven).
[    5.897421] systemd-journald[352]: Received request to flush runtime journal
from PID 1
```

A saída do `dmesg` pode ter centenas de linhas, portanto a listagem anterior contém apenas o trecho que mostra o kernel chamando o gerenciador de serviços `systemd`. Os valores no início das linhas são a quantidade de segundos desde o início do carregamento do kernel.

Nos sistemas baseados em `systemd`, o comando `journalctl` mostra as mensagens de inicialização com as opções `-b`, `--boot`, `-k` ou `--dmesg`. O comando `journalctl --list-boots` mostra uma lista de números de inicialização relativos à inicialização atual, seu hash de identificação e os registros de

data e hora da primeira e última mensagens correspondentes:

```
$ journalctl --list-boots
-4 9e5b3eb4952845208b841ad4dbefa1a6 Thu 2019-10-03 13:39:23 -03-Thu 2019-10-03
13:40:30 -03
-3 9e3d79955535430aa43baa17758f40fa Thu 2019-10-03 13:41:15 -03-Thu 2019-10-03
14:56:19 -03
-2 17672d8851694e6c9bb102df7355452c Thu 2019-10-03 14:56:57 -03-Thu 2019-10-03
19:27:16 -03
-1 55c0d9439fbfb4e85a20a62776d0dbb4d Thu 2019-10-03 19:27:53 -03-Fri 2019-10-04
00:28:47 -03
 0 08fbbebd9f964a74b8a02bb27b200622 Fri 2019-10-04 00:31:01 -03-Fri 2019-10-04
10:17:01 -03
```

Os logs de inicialização anteriores também são preservados nos sistemas baseados no systemd, sendo assim possível inspecionar as mensagens de sessões anteriores do sistema operacional. Se as opções `-b 0` ou `--boot=0` forem usadas, aparecem as mensagens da inicialização atual. As opções `-b -1` ou `--boot=-1` exibem as mensagens da inicialização anterior. As opções `-b -2` ou `--boot=-2` exibem as mensagens da inicialização antes da anterior e assim por diante. O trecho a seguir mostra o kernel chamando o gerenciador de serviços do systemd para o último processo de inicialização:

```
$ journalctl -b 0
oct 04 00:31:01 ubuntu-host kernel: EXT4-fs (sda1): mounted filesystem with ordered
data mode. Opts: (null)
oct 04 00:31:01 ubuntu-host kernel: ip_tables: (C) 2000-2006 Netfilter Core Team
oct 04 00:31:01 ubuntu-host systemd[1]: systemd 237 running in system mode.
oct 04 00:31:01 ubuntu-host systemd[1]: Detected architecture x86-64.
oct 04 00:31:01 ubuntu-host systemd[1]: Set hostname to <torre>.
oct 04 00:31:01 ubuntu-host systemd[1]: Reached target User and Group Name Lookups.
oct 04 00:31:01 ubuntu-host systemd[1]: Created slice System Slice.
oct 04 00:31:01 ubuntu-host systemd[1]: Listening on Journal Audit Socket.
oct 04 00:31:01 ubuntu-host systemd[1]: Listening on Journal Socket.
oct 04 00:31:01 ubuntu-host systemd[1]: Mounting POSIX Message Queue File System...
oct 04 00:31:01 ubuntu-host systemd[1]: Started Read required files in advance.
oct 04 00:31:01 ubuntu-host systemd[1]: Starting Load Kernel Modules...
oct 04 00:31:01 ubuntu-host kernel: EXT4-fs (sda1): re-mounted. Opts:
commit=300,barrier=0,errors=remount-ro
oct 04 00:31:01 ubuntu-host kernel: lp: driver loaded but no devices found
oct 04 00:31:01 ubuntu-host kernel: ppdev: user-space parallel port driver
oct 04 00:31:01 ubuntu-host kernel: parport_pc 00:02: reported by Plug and Play
ACPI
oct 04 00:31:01 ubuntu-host kernel: parport0: PC-style at 0x378 (0x778), irq 7, dma
```

```
3 [PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
oct 04 00:31:01 ubuntu-host kernel: lp0: using parport0 (interrupt-driven).
oct 04 00:31:01 ubuntu-host systemd-journald[352]: Journal started
oct 04 00:31:01 ubuntu-host systemd-journald[352]: Runtime journal
(/run/log/journal/abb765408f3741ae9519ab3b96063a15) is 4.9M, max 39.4M, 34.5M free.
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'lp'
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'ppdev'
oct 04 00:31:01 ubuntu-host systemd-modules-load[335]: Inserted module 'parport_pc'
oct 04 00:31:01 ubuntu-host systemd[1]: Starting Flush Journal to Persistent
Storage...
```

As mensagens sobre a inicialização e outras emitidas pelo sistema operacional são armazenadas em arquivos dentro do diretório `/var/log/`. Se ocorrer um erro crítico e o sistema operacional for incapaz de continuar o processo de inicialização depois de carregar o kernel e o initramfs, uma mídia de inicialização alternativa pode ser usada para iniciar o sistema e acessar o arquivo de sistemas correspondente. Depois disso, os arquivos em `/var/log/` podem ser consultados em busca das razões causaram a interrupção do processo de inicialização. As opções `-D` ou `--directory` do comando `journalctl` servem para ler mensagens de log em diretórios diferentes de `/var/log/journal/`, que é a localização padrão para as mensagens de log do `systemd`. Como as mensagens de log do sistema não são armazenadas em texto puro, o comando `journalctl` é necessário para que fiquem legíveis.

Exercícios Guiados

1. Em uma máquina equipada com firmware BIOS, onde está localizado o binário do bootstrap?

2. O firmware UEFI suporta recursos estendidos fornecidos por programas externos, chamados aplicativos EFI. Esses aplicativos, no entanto, têm seu próprio local especial. Em que lugar do sistema localizam-se os aplicativos?

3. Os gerenciadores de inicialização permitem a passagem de parâmetros personalizados do kernel antes de carregá-lo. Suponha que o sistema não possa inicializar devido a uma localização incorreta do sistema de arquivos raiz. Como o sistema de arquivos raiz correto, localizado em /dev/sda3, seria fornecido como parâmetro para o kernel?

4. O processo de inicialização de uma máquina Linux termina com a seguinte mensagem:

ALERT! /dev/sda3 does not exist. Dropping to a shell!

Qual a causa provável desse problema?

Exercícios Exploratórios

1. O carregador de inicialização apresenta uma lista de sistemas operacionais a escolher quando houver mais de um sistema operacional instalado na máquina. No entanto, um sistema operacional recém-instalado pode sobreescriver o MBR do disco rígido, apagando o primeiro estágio do gerenciador de inicialização e tornando o outro sistema operacional inacessível. Por que isso não aconteceria em uma máquina equipada com um firmware UEFI?

2. Qual é uma consequência comum de se instalar um kernel personalizado sem fornecer uma imagem initramfs apropriada?

3. O log de inicialização tem centenas de linhas, portanto a saída do comando `dmesg` é frequentemente canalizada para um comando de paginação—como o comando `less`—para facilitar a leitura. Qual opção do `dmesg` faz automaticamente a paginação da saída, eliminando a necessidade de usar explicitamente um comando de paginação?

4. Um disco rígido contendo todo o sistema de arquivos de uma máquina offline foi removido e conectado a uma máquina operacional como drive secundário. Supondo que seu ponto de montagem seja `/mnt/hd`, como o `journalctl` seria usado para inspecionar o conteúdo dos arquivos de diário localizados em `/mnt/hd/var/log/journal/`?

Resumo

Esta lição aborda a sequência de inicialização em um sistema Linux padrão. O conhecimento adequado de como funciona o processo de inicialização de um sistema Linux ajuda a evitar erros que podem tornar o sistema inacessível. A lição tratou dos seguintes tópicos:

- Diferença entre os métodos de inicialização da BIOS e da UEFI.
- Estágios típicos da inicialização do sistema.
- Recuperação de mensagens de inicialização.

Os comandos e procedimentos abordados foram:

- Parâmetros comuns do kernel.
- Comandos para ler mensagens de inicialização: `dmesg` e `journalctl`.

Respostas aos Exercícios Guiados

- Em uma máquina equipada com firmware BIOS, onde está localizado o binário do bootstrap?

No MBR do primeiro dispositivo de armazenamento, como definido no utilitário de configuração da BIOS.

- O firmware UEFI suporta recursos estendidos fornecidos por programas externos, chamados aplicativos EFI. Esses aplicativos, no entanto, têm seu próprio local especial. Em que lugar do sistema localizam-se os aplicativos?

Os aplicativos EFI são armazenados na EFI System Partition (ESP), localizada em qualquer bloco de armazenamento disponível com um sistema de arquivos compatível (geralmente um sistema de arquivos FAT32).

- Os gerenciadores de inicialização permitem a passagem de parâmetros personalizados do kernel antes de carregá-lo. Suponha que o sistema não possa inicializar devido a uma localização incorreta do sistema de arquivos raiz. Como o sistema de arquivos raiz correto, localizado em `/dev/sda3`, seria fornecido como parâmetro para o kernel?

O parâmetro `root` deve ser usado, como em `root=/dev/sda3`.

- O processo de inicialização de uma máquina Linux termina com a seguinte mensagem:

`ALERT! /dev/sda3 does not exist. Dropping to a shell!`

Qual a causa provável desse problema?

O kernel não encontrou o dispositivo `/dev/sda3`, informado como o sistema de arquivo raiz.

Respostas aos Exercícios Exploratórios

- O carregador de inicialização apresenta uma lista de sistemas operacionais a escolher quando houver mais de um sistema operacional instalado na máquina. No entanto, um sistema operacional recém-instalado pode sobreescriver o MBR do disco rígido, apagando o primeiro estágio do gerenciador de inicialização e tornando o outro sistema operacional inacessível. Por que isso não aconteceria em uma máquina equipada com um firmware UEFI?

As máquinas UEFI não usam o MBR do disco rígido para armazenar o primeiro estágio do gerenciador de inicialização.

- Qual é uma consequência comum de se instalar um kernel personalizado sem fornecer uma imagem initramfs apropriada?

O sistema de arquivos raiz pode ficar inacessível se seu tipo tiver sido compilado como um módulo externo do kernel.

- O log de inicialização tem centenas de linhas, portanto a saída do comando `dmesg` é frequentemente canalizada para um comando de paginação—como o comando `less`—para facilitar a leitura. Qual opção do `dmesg` faz automaticamente a paginação da saída, eliminando a necessidade de usar explicitamente um comando de paginação?

Os comandos `dmesg -H` ou `dmesg --human` habilitam a paginação por padrão.

- Um disco rígido contendo todo o sistema de arquivos de uma máquina offline foi removido e conectado a uma máquina operacional como drive secundário. Supondo que seu ponto de montagem seja `/mnt/hd`, como o `journalctl` seria usado para inspecionar o conteúdo dos arquivos de diário localizados em `/mnt/hd/var/log/journal/`?

Com os comandos `journalctl -D /mnt/hd/var/log/journal` ou `journalctl --directory=/mnt/hd/var/log/journal`



101.3 Alternar runlevels/boot targets, desligar e reiniciar o sistema

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 101.3

Peso

3

Áreas chave de conhecimento

- Definir o runlevel padrão e o alvo de boot padrão.
- Alternar entre os runlevels/alvos de boot, incluindo o modo single user (usuário único).
- Desligar e reiniciar através da linha de comando.
- Alertar os usuários antes de mudar o runlevel/alvo de boot ou outro evento de sistema que acarrete uma mudança significativa.
- Terminar apropriadamente os processos.
- Noções de acpid.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- /etc/inittab
- shutdown
- init
- /etc/init.d/
- telinit
- systemd

- `systemctl`
- `/etc/systemd/`
- `/usr/lib/systemd/`
- `wall`



101.3 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	101 Arquitetura do sistema
Objetivo:	101.3 Alterar níveis de execução/destinos de inicialização e desligar ou reiniciar o sistema
Lição:	1 de 1

Introdução

Uma característica comum entre os sistemas operacionais que seguem os princípios de design do Unix é o emprego de processos separados para controlar funções distintas do sistema. Esses processos, chamados *daemons* (ou, mais geralmente, *serviços*), também são responsáveis pelos recursos estendidos subjacentes ao sistema operacional, como serviços de aplicativos de rede (servidor HTTP, compartilhamento de arquivos, email...), bancos de dados, configuração sob demanda etc. Embora o Linux utilize um kernel monolítico, muitos aspectos de baixo nível do sistema operacional são afetados por daemons, como o balanceamento de carga e a configuração do firewall.

Os daemons que devem estar ativos dependem da finalidade do sistema. O conjunto de daemons ativos também deve ser modificável em tempo de execução, para que os serviços possam ser iniciados e interrompidos sem a necessidade de se reiniciar o sistema inteiro. Para resolver esse problema, todas as principais distribuições Linux oferecem algum tipo de utilitário de gerenciamento de serviços para controlar o sistema.

Os serviços podem ser controlados por scripts do shell ou por um programa e seus arquivos de

configuração. O primeiro método é implementado pelo padrão *SysVinit*, também conhecido como *System V* ou apenas *SysV*. O segundo método é implementado por *systemd* e *Upstart*. Historicamente, os gerenciadores de serviços baseados em *SysV* eram os mais utilizados pelas distribuições Linux. Hoje, os gerenciadores de serviços baseados no *systemd* são os mais frequentes na maioria das distribuições Linux. O gerenciador de serviços é o primeiro programa lançado pelo kernel durante o processo de inicialização, portanto seu PID (número de identificação do processo) é sempre 1.

SysVinit

Um gerenciador de serviços baseado no padrão *SysVinit* fornece conjuntos predefinidos de estados do sistema, chamados *níveis de execução*, além dos arquivos correspondentes de script dos serviços a serem executados. Os níveis de execução são numerados de 0 a 6, geralmente atribuídos às seguintes finalidades:

Nível de execução 0

Encerramento do sistema.

Nível de execução 1, s ou single

Modo de usuário único, sem rede e outros recursos não-essenciais (modo de manutenção).

Nível de execução 2, 3 ou 4

Modo multiusuário. Os usuários podem se logar via console ou pela rede. Os níveis de execução 2 e 4 não são usados com frequência.

Nível de execução 5

Modo multiusuário. Equivale ao nível 3, mas o login em modo gráfico.

Nível de execução 6

Reinicialização do sistema.

O programa responsável pelo gerenciamento de níveis de execução e daemons/recursos associados é `/sbin/init`. Durante a inicialização do sistema, o programa `init` identifica o nível de execução solicitado, definido por um parâmetro do kernel ou no arquivo `/etc/inittab`, e carrega os scripts associados listados ali para o nível de execução especificado. Cada nível de execução pode ter diversos arquivos de serviço associados, geralmente scripts no diretório `/etc/init.d/`. Como nem todos os níveis de execução se equivalem nas diferentes distribuições Linux, uma breve descrição do objetivo do nível de execução também pode ser encontrada nas distribuições baseadas em *SysV*.

A sintaxe do arquivo `/etc/inittab` usa o seguinte formato:

id:runlevels:action:process

id é um nome genérico de até quatro caracteres usado para identificar a entrada. O item **runlevels** é uma lista de números dos níveis de execução nos quais uma ação especificada deve ser executada. O termo **action** define como **init** executará o processo indicado pelo termo **process**. As ações disponíveis são:

boot

O processo será executado durante a inicialização do sistema. O campo **runlevels** é ignorado.

bootwait

O processo será ignorado durante a inicialização do sistema e o **init** aguardará sua conclusão para continuar. O campo **runlevels** é ignorado.

sysinit

O processo será executado após a inicialização do sistema, qualquer que seja o nível de execução. O campo **runlevels** é ignorado.

wait

O processo será executado nos níveis de execução dados e **init** aguardará sua conclusão para continuar.

respawn

O processo será reiniciado caso seja encerrado.

ctrlaltdel

O processo será executado quando o processo **init** receber o sinal SIGINT, disparado quando o atalho de teclado **Ctrl + Alt + Del** for pressionado.

O nível de execução padrão — que será escolhido se nenhum outro for fornecido como parâmetro do kernel — também é definido em **/etc/inittab**, na entrada **id:x:initdefault**. **x** é o número do nível de execução padrão. Esse número nunca deve ser **0** ou **6**, pois isso faria com que o sistema desligasse ou reiniciasse ao final do processo de inicialização. Um arquivo **/etc/inittab** típico é mostrado abaixo:

```
# Default runlevel
id:3:initdefault:

# Configuration script executed during boot
```

```

si::sysinit:/etc/init.d/rcS

# Action taken on runlevel S (single user)
~:S:wait:/sbin/sulogin

# Configuration for each execution level
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# Action taken upon ctrl+alt+del keystroke
ca::ctrlaltdel:/sbin/shutdown -r now

# Enable consoles for runlevels 2 and 3
1:23:respawn:/sbin/getty tty1 VC linux
2:23:respawn:/sbin/getty tty2 VC linux
3:23:respawn:/sbin/getty tty3 VC linux
4:23:respawn:/sbin/getty tty4 VC linux

# For runlevel 3, also enable serial
# terminals ttyS0 and ttyS1 (modem) consoles
S0:3:respawn:/sbin/getty -L 9600 ttyS0 vt320
S1:3:respawn:/sbin/mgetty -x0 -D ttyS1

```

O comando `telinit q` deve ser executado a cada vez que o arquivo `/etc/inittab` é modificado. O argumento `q` (ou `Q`) pede que o init recarregue sua configuração. Essa etapa é importante para evitar uma parada do sistema devido a uma configuração incorreta em `/etc/inittab`.

Os scripts usados pelo `init` para configurar cada nível de execução ficam armazenados no diretório `/etc/init.d/`. Cada nível de execução tem um diretório associado em `/etc/`, chamado `/etc/rc0.d/`, `/etc/rc1.d/`, `/etc/rc2.d/`, etc., com os scripts que devem ser executados quando cada nível de execução é iniciado. Como o mesmo script pode ser usado por diferentes níveis de execução, os arquivos nesses diretórios são apenas links simbólicos para os scripts reais em `/etc/init.d/`. Além disso, a primeira letra do nome do arquivo do link no diretório do nível de execução indica se o serviço deve ser iniciado ou encerrado no nível de execução correspondente. Um nome de arquivo iniciado com a letra `K` determina que o serviço será encerrado ao entrar no nível de execução (`kill`). Se começar com a letra `S`, o serviço será iniciado ao ingressar no nível de execução (`start`). O diretório `/etc/rc1.d/`, por exemplo, terá muitos links para scripts de rede iniciados com a letra `K`, considerando que o nível de execução 1 é o nível do usuário único, sem conectividade de rede.

O comando `runlevel` mostra o atual nível de execução do sistema. Esse comando exibe dois valores: o primeiro é o nível de execução anterior, o segundo o atual:

```
$ runlevel
N 3
```

A letra `N` na saída mostra que o nível de execução não mudou desde a última inicialização. No exemplo, `runlevel 3` é o nível de execução atual do sistema.

O mesmo programa `init` pode ser usado para alternar entre níveis de execução em um sistema em execução sem a necessidade de reiniciar. O comando `telinit` também permite alternar entre níveis de execução. Por exemplo, os comandos `telinit 1`, `telinit s` ou `telinit S` mudarão o sistema para o nível de execução 1.

systemd

Atualmente, o `systemd` é o conjunto de ferramentas mais usado para gerenciar recursos e serviços do sistema, chamados de *unidades* (*units*) pelo `systemd`. Uma unidade consiste em um nome, um tipo e um arquivo de configuração correspondente. Por exemplo, a unidade para um processo de servidor `httpd` (como o servidor web Apache) será `httpd.service` nas distribuições baseadas no Red Hat e seu arquivo de configuração também será chamado `httpd.service` (nas distribuições baseadas em Debian, essa unidade é chamada `apache2.service`).

Existem sete tipos distintos de unidades `systemd`:

`service`

O tipo de unidade mais comum, para recursos ativos do sistema que podem ser iniciados, interrompidos e recarregados.

`socket`

O tipo de unidade `socket` pode ser um socket de sistema de arquivos ou um socket de rede. Todas as unidades `socket` possuem uma unidade de serviço correspondente, carregada quando o socket recebe uma solicitação.

`device`

Uma unidade de dispositivo está associada a um dispositivo de hardware identificado pelo kernel. Um dispositivo só será considerado como uma unidade `systemd` se existir uma regra `udev` para isso. Uma unidade de dispositivo pode ser usada para resolver dependências de configuração quando determinado hardware é detectado, uma vez que as propriedades da regra `udev` podem ser usadas como parâmetros para a unidade de dispositivo.

mount

Uma unidade de montagem é uma definição de ponto de montagem no sistema de arquivos, semelhante a uma entrada em /etc/fstab.

automount

Uma unidade de montagem automática também é uma definição de ponto de montagem no sistema de arquivos, mas nesse caso montada automaticamente. Cada unidade de montagem automática possui uma unidade de montagem correspondente, que é iniciada quando o ponto de montagem automática é acessado.

target

Uma unidade de destino é um agrupamento de outras unidades, gerenciadas como uma única unidade.

snapshot

Uma unidade snapshot é um estado salvo do gerenciador do systemd (não disponível em todas as distribuições Linux).

O principal comando para controlar as unidades do systemd é `systemctl`. O comando `systemctl` é usado para executar todas as tarefas relacionadas à ativação, desativação, execução, interrupção e monitoramento da unidade, entre outras. Para uma unidade fictícia chamada `unit.service`, por exemplo, as ações mais comuns do `systemctl` serão:

`systemctl start unit.service`

Inicia `unit`.

`systemctl stop unit.service`

Interrompe `unit`.

`systemctl restart unit.service`

Reinicia `unit`.

`systemctl status unit.service`

Mostra o estado de `unit`, incluindo se está ou não em execução.

`systemctl is-active unit.service`

Exibe *active* se `unit` estiver rodando, ou *inactive* se não estiver.

`systemctl enable unit.service`

Habilita `unit`, ou seja, `unit` será carregado durante a inicialização do sistema.

systemctl disable unit.service

unit não será iniciado com o sistema.

systemctl is-enabled unit.service

Verifica se unit é iniciado com o sistema. A resposta é armazenada na variável `$?`. O valor `0` indica que unit inicia com o sistema e o valor `1` indica que não.

As instalações mais recentes do systemd listam, na verdade, a configuração de uma unidade para o tempo de inicialização. Por exemplo:

NOTE

```
$ systemctl is-enabled apparmor.service
enabled
```

Se não houver outras unidades com o mesmo nome no sistema, o sufixo após o ponto poderá ser descartado. Se, por exemplo, houver apenas uma unidade `httpd` do tipo service, somente `httpd` bastará como parâmetro de unidade para `systemctl`.

O comando `systemctl` também pode controlar os *destinos do sistema*. A unidade `multi-user.target`, por exemplo, combina todas as unidades exigidas pelo ambiente do sistema multiusuário. É semelhante ao nível de execução número 3 em um sistema que utiliza o SysV.

O comando `systemctl isolate` alterna entre diferentes destinos. Assim, para alternar manualmente para o destino `multi-user`:

```
# systemctl isolate multi-user.target
```

Existem destinos correspondentes aos níveis de execução do SysV, desde `runlevel0.target` até `runlevel6.target`. Todavia, o systemd não usa o arquivo `/etc/inittab`. Para alterar o destino padrão do sistema, a opção `systemd.unit` pode ser adicionada à lista de parâmetros do kernel. Por exemplo, para usar `multi-user.target` como destino padrão, o parâmetro do kernel deve ser `systemd.unit=multi-user.target`. Todos os parâmetros do kernel podem ser tornados persistentes alterando-se a configuração do gerenciador de inicialização.

Outra maneira de alterar o destino padrão é modificar o link simbólico `/etc/systemd/system/default.target` para que ele aponte para o destino desejado. A redefinição do link pode ser feita com o próprio comando `systemctl`:

```
# systemctl set-default multi-user.target
```

Da mesma forma, podemos determinar o destino de inicialização padrão do sistema com o seguinte comando:

```
$ systemctl get-default  
graphical.target
```

Como nos sistemas que adotam o SysV, o destino padrão nunca deve apontar para `shutdown.target`, pois ele corresponde ao nível de execução 0 (encerramento).

Os arquivos de configuração associados a cada unidade ficam no diretório `/lib/systemd/system/`. O comando `systemctl list-unit-files` lista todas as unidades disponíveis e mostra se elas estão habilitadas para iniciar quando o sistema é inicializado. A opção `--type` seleciona apenas as unidades para um tipo determinado, como em `systemctl list-unit-files --type=service` e `systemctl list-unit-files --type=target`.

As unidades ativas ou as que estiveram ativas durante a sessão atual do sistema podem ser listadas com o comando `systemctl list-units`. Como no caso da opção `list-unit-files`, o comando `systemctl list-units --type=service` seleciona somente as unidades de tipo `service` e o comando `systemctl list-units --type=target` seleciona somente as unidades do tipo `target`.

O `systemd` também é responsável por acionar e responder a eventos relacionados ao consumo de energia. O comando `systemctl suspend` coloca o sistema no modo de baixo consumo de energia, mantendo os dados atuais na memória. O comando `systemctl hibernate` copia todos os dados da memória no disco, para que o estado atual do sistema possa ser recuperado após o desligamento. As ações associadas a esses eventos são definidas no arquivo `/etc/systemd/logind.conf` ou em arquivos separados dentro do diretório `/etc/systemd/logind.conf.d/`. No entanto, esse recurso do `systemd` pode ser usado apenas quando não houver outro gerenciador de energia em execução no sistema, como o daemon `acpid`. O daemon `acpid` é o principal gerenciador de energia do Linux e permite ajustes mais refinados das ações após eventos relacionados ao consumo de energia, como fechar a tampa do laptop, bateria fraca ou níveis de carga da bateria.

Upstart

Os scripts de inicialização usados pelo Upstart estão localizados no diretório `/etc/init/`. Os serviços do sistema podem ser listados com o comando `initctl list`, que também mostra o estado atual dos serviços e, se disponível, seu número PID.

```
# initctl list  
avahi-cups-reload stop/waiting  
avahi-daemon start/running, process 1123
```

```
mountall-net stop/waiting
mountnfs-bootclean.sh start/running
nmbd start/running, process 3085
passwd stop/waiting
rc stop/waiting
rsyslog start/running, process 1095
tty4 start/running, process 1761
udev start/running, process 1073
upstart-udev-bridge start/running, process 1066
console-setup stop/waiting
irqbalance start/running, process 1842
plymouth-log stop/waiting
smbd start/running, process 1457
tty5 start/running, process 1764
failsafe stop/waiting
```

Cada uma das ações do Upstart tem seu próprio comando independente. Por exemplo, o comando `start` pode ser usado para iniciar um sexto terminal virtual:

```
# start tty6
```

O estado atual de um recurso pode ser verificado com o comando `status`:

```
# status tty6
tty6 start/running, process 3282
```

E a interrupção de um serviço é feita com o comando `stop`:

```
# stop tty6
```

O Upstart não usa o arquivo `/etc/inittab` para definir os níveis de execução, mas os comandos legados `runlevel` e `telinit` ainda podem ser usados para verificar e alternar entre níveis de execução.

NOTE

O Upstart foi desenvolvido para a distribuição Ubuntu Linux para facilitar a inicialização paralela dos processos. O Ubuntu parou de usar o Upstart em 2015, quando migrou do Upstart para o systemd.

Desligar e reiniciar

Um comando muito tradicional usado para desligar ou reiniciar o sistema é o `shutdown`. O comando `shutdown` adiciona funções extras ao processo de desligamento: ele notifica automaticamente todos os usuários conectados com uma mensagem em suas sessões do shell e impede novos logins. O comando `shutdown` atua como intermediário nos procedimentos do SysV ou do `systemd`, ou seja, executa a ação solicitada chamando a ação correspondente no gerenciador de serviços adotado pelo sistema.

Após o `shutdown` ser executado, todos os processos recebem o sinal `SIGTERM`, seguido pelo sinal `SIGKILL`, e o sistema é encerrado ou muda seu nível de execução. Por padrão, quando as opções `-h` ou `-r` não são usadas, o sistema alterna para o nível de execução 1, ou seja, o modo de usuário único. Para alterar as opções padrão de `shutdown`, o comando deve ser executado com a seguinte sintaxe:

```
$ shutdown [option] time [message]
```

Somente o parâmetro `time` é obrigatório. Esse parâmetro define quando a ação solicitada será executada, aceitando os seguintes formatos:

hh:mm

Este formato especifica o tempo de execução em horas e minutos.

+m

Este formato especifica quantos minutos esperar antes da execução.

now ou +0

Este formato determina a execução imediata.

O parâmetro `message` é o texto de aviso enviado a todas as sessões de terminal dos usuários logados.

A implementação do SysV permite limitar os usuários que poderão reiniciar a máquina pressionando `Ctrl + Alt + Del`. Para isso, incluímos a opção `-a` no comando `shutdown` presente na linha referente a `ctrlaltdel` no arquivo `/etc/inittab`. Ao fazer isso, somente usuários cujos nomes de usuário constem do arquivo `/etc/shutdown.allow` poderão reiniciar o sistema com o atalho de teclado `Ctrl + Alt + Del`.

O comando `systemctl` também pode ser usado para desligar ou reiniciar a máquina em sistemas que empregam o `systemd`. Para reiniciar o sistema, usamos o comando `systemctl reboot`. Para desligar o sistema, o comando é `systemctl poweroff`. Ambos os comandos requerem privilégios de root para serem executados, pois usuários comuns não podem realizar esses procedimentos.

Algumas distribuições Linux vinculam poweroff e reboot a systemctl como comandos individuais. Por exemplo:

NOTE

```
$ sudo which poweroff  
/usr/sbin/poweroff  
$ sudo ls -l /usr/sbin/poweroff  
lrwxrwxrwx 1 root root 14 Aug 20 07:50 /usr/sbin/poweroff ->  
/bin/systemctl
```

Nem todas as atividades de manutenção exigem que o sistema seja desligado ou reiniciado. No entanto, quando é necessário alterar o estado do sistema para o modo de usuário único, é importante avisar os usuários conectados para que não sejam prejudicados pelo encerramento abrupto de suas atividades.

Como no caso do comando shutdown ao desligar ou reiniciar o sistema, o comando wall pode enviar uma mensagem para as sessões de terminal de todos os usuários logados. Para isso, o administrador do sistema só precisa fornecer um arquivo ou escrever diretamente a mensagem como um parâmetro para o comando wall.

Exercícios Guiados

1. Como o comando `telinit` pode ser usado para reiniciar o sistema?

2. O que acontece com os serviços relacionados ao arquivo `/etc/rc1.d/K90network` quando o sistema entra no nível de execução 1?

3. Usando o comando `systemctl`, como um usuário pode verificar se a unidade `sshd.service` está em execução?

4. Em um sistema baseado em `systemd`, qual comando deve ser executado para ativar a unidade `sshd.service` durante a inicialização do sistema?

Exercícios Exploratórios

1. Em um sistema baseado em SysV, suponha que o nível de execução padrão definido em `/etc/inittab` seja 3, mas o sistema sempre inicia no nível de execução 1. Qual é a causa provável disso?

2. Embora o arquivo `/sbin/init` possa ser encontrado nos sistemas baseados em systemd, ele é apenas um link simbólico para outro arquivo executável. Nesses sistemas, qual o arquivo apontado por `/sbin/init`?

3. Como se verifica o destino padrão do sistema em um sistema baseado em systemd?

4. Como poderíamos cancelar uma reinicialização do sistema programada com o comando `shutdown`?

Resumo

Esta lição aborda os principais utilitários usados como gerenciadores de serviços pelas distribuições Linux. Os utilitários SysVinit, systemd e Upstart têm sua abordagem própria para controlar serviços e estados do sistema. A lição inclui os seguintes tópicos:

- O que são os serviços do sistema e qual seu papel no sistema operacional.
- Conceitos e utilização básica dos comandos do SysVinit, systemd e Upstart commands.
- Como iniciar, interromper e reiniciar apropriadamente os serviços do sistema e o próprio sistema.

Os comandos e procedimentos abordados foram:

- Comandos e arquivos relacionados ao SysVinit, como `init`, `/etc/inittab` e `telinit`.
- O comando principal do systemd: `systemctl`.
- Comandos do Upstart: `initctl`, `status`, `start`, `stop`.
- Comandos tradicionais de gerenciamento do sistema, como `shutdown` e o comando `wall`.

Respostas aos Exercícios Guiados

1. Como o comando `telinit` pode ser usado para reiniciar o sistema?

O comando `telinit 6` alterna para o nível de execução 6, ou seja, reinicia o sistema.

2. O que acontece com os serviços relacionados ao arquivo `/etc/rc1.d/K90network` quando o sistema entra no nível de execução 1?

Como mostra a letra K no início do nome do arquivo, os serviços relacionados serão interrompidos.

3. Usando o comando `systemctl`, como um usuário pode verificar se a unidade `sshd.service` está em execução?

Com o comando `systemctl status sshd.service` ou `systemctl is-active sshd.service`.

4. Em um sistema baseado em `systemd`, qual comando deve ser executado para ativar a unidade `sshd.service` durante a inicialização do sistema?

O comando `systemctl enable sshd.service`, executado pelo root.

Respostas aos Exercícios Exploratórios

1. Em um sistema baseado em SysV, suponha que o nível de execução padrão definido em `/etc/inittab` seja 3, mas o sistema sempre inicia no nível de execução 1. Qual é a causa provável disso?

Os parâmetros `1` ou `S` podem estar presentes na lista de parâmetros do kernel.

2. Embora o arquivo `/sbin/init` possa ser encontrado nos sistemas baseados em systemd, ele é apenas um link simbólico para outro arquivo executável. Nesses sistemas, qual o arquivo apontado por `/sbin/init`?

O binário principal do systemd: `/lib/systemd/systemd`.

3. Como se verifica o destino padrão do sistema em um sistema baseado em systemd?

O link simbólico `/etc/systemd/system/default.target` aponta para o arquivo da unidade definido como destino padrão. Também é possível usar o comando `systemctl get-default`.

4. Como poderíamos cancelar uma reinicialização do sistema programada com o comando `shutdown`?

Usaríamos o comando `shutdown -c`.



Tópico 102: Instalação do Linux e administração de Pacotes



102.1 Dimensionar partições de disco

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 102.1

Peso

2

Áreas chave de conhecimento

- Distribuir os sistemas de arquivos e o espaço de swap para separar partições ou discos.
- Adaptar o projeto para o uso pretendido do sistema.
- Garantir que a partição /boot esteja em conformidade com os requisitos de arquitetura de hardware para a inicialização.
- Conhecimento das características básicas do LVM.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- Sistema de arquivos raiz / (root)
- Sistema de arquivos /var
- Sistema de arquivos /home
- Sistema de arquivos /boot
- Partição de sistema EFI (ESP)
- Espaço de swap
- Pontos de montagem
- Partições



102.1 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	102 Instalação do Linux e gerenciamento de pacotes
Objetivo:	102.1 Definir o esquema de partições do disco rígido
Lição:	1 de 1

Introdução

Para concluir este objetivo com sucesso, é necessário entender a relação entre *discos*, *partições*, *sistemas de arquivo* e *volumes*.

Pense em um disco (ou *dispositivo de armazenamento*, já que os dispositivos modernos não contêm nenhum “disco”) como um “receptáculo físico” para seus dados.

Antes de um disco poder ser usado por um computador, ele precisa ser particionado. Uma partição é um subconjunto lógico do disco físico, uma espécie de “cerca” lógica. O particionamento é uma maneira de “compartimentar” as informações armazenadas no disco, separando, por exemplo, os dados do sistema operacional dos dados do usuário.

Todo disco precisa de pelo menos uma partição, mas é possível definir várias partições se necessário, e as informações a respeito delas 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 detalhes adicionais sobre cada partição.

Dentro de cada partição existe um sistema de arquivos. O sistema de arquivos descreve a maneira como as informações estão de fato armazenadas no disco; por exemplo, o modo como os diretórios são organizados, qual a relação entre eles, onde estão os dados para cada arquivo etc.

As partições não podem abranger vários discos. Porém, usando o *Gerenciador de Volumes Lógicos* (Logic Volume Manager ou LVM), podemos combinar várias partições, mesmo entre discos, para formar um único volume lógico.

Os volumes lógicos abstraem as limitações dos dispositivos físicos e permitem trabalhar com “pools” de espaço em disco que podem ser combinados ou distribuídos de maneira muito mais flexível do que as partições tradicionais permitiriam. O LVM é útil em situações em é necessário adicionar mais espaço a uma partição sem precisar migrar os dados para um dispositivo maior.

Neste objetivo, você aprenderá a projetar um esquema de particionamento de disco para um sistema Linux, alocando sistemas de arquivos e espaços virtuais de troca para separar partições ou discos quando necessário.

A maneira de *criar* e *gerenciar* partições e sistemas de arquivos será discutida em outras lições. Aqui, nos concentraremos em dar uma visão geral do LVM.

Pontos de montagem

Antes que um sistema de arquivos possa ser acessado no Linux, ele precisa ser *montado*. Isso significa vincular o sistema de arquivos a um ponto específico na árvore de diretórios do sistema, chamado *ponto de montagem*.

Quando montado, o conteúdo do sistema de arquivos estará disponível no ponto de montagem. Por exemplo, imagine que você tenha uma partição com os dados pessoais de seus usuários (o diretório pessoal deles), contendo os diretórios `/john`, `/jack` e `/carol`. Quando montados em `/home`, o conteúdo desses diretórios estará disponível em `/home/john`, `/home/jack` e `/home/carol`.

O ponto de montagem deve existir antes de o sistema de arquivos ser montado. Não é possível montar uma partição em `/mnt/userdata` se esse diretório não existir. No entanto, se o diretório existir e contiver arquivos, esses arquivos ficarão indisponíveis até você desmontar o sistema de arquivos. Ao listar o conteúdo do diretório, veremos os arquivos armazenados no sistema de arquivos montado, não o conteúdo original do diretório.

Os sistemas de arquivos podem ser montados em qualquer lugar que você desejar. No entanto, certas práticas são recomendadas para facilitar a administração do sistema.

Tradicionalmente, todos os dispositivos externos eram montados no diretório `/mnt`. Ali existia uma série de *pontos de ancoragem* pré-configurados para os dispositivos mais comuns, como drives de

CD-ROM (/mnt/cdrom) e de disquete (/mnt/floppy).

Ele foi substituído por /media, que agora é o ponto de montagem padrão para qualquer mídia removível (como discos externos, drives flash USB, leitores de cartão de memória, discos ópticos 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 for preciso montar *manualmente* um sistema de arquivos, é recomendável montá-lo em /mnt. Os comandos específicos para controlar a montagem e desmontagem de sistemas de arquivos no Linux serão discutidos em outra lição.

Mantendo as coisas separadas

No Linux, existem alguns diretórios que é melhor manter em partições separadas. Há muitas razões para isso: por exemplo, ao manter os arquivos relacionados ao carregador de inicialização (armazenado em /boot) em uma partição boot, garantimos que o sistema ainda poderá inicializar em caso de falha no sistema de arquivos raiz.

Também é mais fácil reinstalar o sistema sem o risco de afetar os dados dos usuários se os diretórios pessoais (em /home) estiverem em uma partição separada. Da mesma forma, manter os dados relacionados a um servidor web ou banco de dados (normalmente em /var) em uma partição separada (ou mesmo um disco à parte) facilita a administração do sistema caso seja necessário adicionar mais espaço em disco para esses casos de uso.

Há também razões de desempenho para manter determinados diretórios em partições separadas. Você pode querer manter o sistema de arquivos raiz (/) em uma unidade SSD rápida, e diretórios maiores como /home e /var em discos rígidos mais lentos, que oferecem muito mais espaço por um preço bem menor.

A partição de inicialização (/boot)

A partição de inicialização contém arquivos usados pelo gerenciador de inicialização para carregar o sistema operacional. Nos sistemas Linux, o gerenciador de inicialização costuma ser o GRUB2 ou, em sistemas mais antigos, o GRUB Legacy. A partição geralmente é montada em /boot e seus arquivos são armazenados em /boot/grub.

Tecnicamente, uma partição de inicialização não é necessária, pois na maioria dos casos o GRUB

pode montar a partição raiz (/) e carregar os arquivos a partir de um diretório /boot separado.

No entanto, uma partição de inicialização separada pode ser interessante, seja por segurança (garantindo que o sistema seja inicializado mesmo em caso de falha no sistema de arquivos raiz), ou se você desejar usar um sistema de arquivos que o carregador de inicialização não pode entender na partição raiz, ou se usar um método não suportado de criptografia ou compactação.

A partição de inicialização geralmente é a primeira partição no disco. Isso ocorre porque a IBM PC BIOS original endereçava os discos usando *cilindros*, *cabeças* e *setores* (CHS), com um máximo de 1024 cilindros, 256 cabeças e 63 setores, resultando em um tamanho máximo de disco de 528 MB (504 MB no MS-DOS). Isso significa que qualquer coisa além dessa marca não estaria acessível em sistemas legados, a menos que um esquema de endereçamento de disco diferente (como o *Endereçamento de bloco lógico*, LBA) fosse usado.

Portanto, para obter compatibilidade máxima, a partição de inicialização geralmente está localizada no início do disco e termina antes do cilindro 1024 (528 MB), garantindo que, aconteça o que acontecer, a máquina sempre poderá carregar o kernel.

Como a partição de inicialização armazena apenas os arquivos necessários para o carregador de inicialização, o disco RAM inicial e as imagens do kernel, ela pode ser bem pequena para os padrões atuais. Um bom tamanho é de cerca de 300 MB.

A partição do sistema EFI (ESP)

A *partição do sistema EFI* (ESP) é usada por máquinas baseadas na UEFI (*Unified Extensible Firmware Interface*) para armazenar gerenciadores de inicialização e imagens do kernel para os sistemas operacionais instalados.

Essa partição é formatada em um sistema de arquivos baseado em FAT. Em um disco particionado com uma tabela de partição GUID, ela possui o identificador global único C12A7328-F81F-11D2-BA4B-00A0C93EC93B. Se o disco tiver sido formatado no esquema de particionamento do MBR, o ID da partição será 0xEF.

Em máquinas com Microsoft Windows, essa partição geralmente é a primeira no disco, embora isso não seja obrigatório. O ESP é criado (ou preenchido) pelo sistema operacional após a instalação e, em um sistema Linux, é montado em /boot/efi.

A partição /home

Cada usuário do sistema possui um diretório inicial para armazenar arquivos pessoais e preferências. A maioria deles está localizada em /home. O diretório inicial costuma ter o mesmo nome que o usuário; assim, o usuário John teria seu diretório em /home/john.

Porém, há exceções. Por exemplo, o diretório inicial do usuário raiz é `/root` e alguns serviços do sistema podem ter usuários associados com diretórios pessoais em outros lugares.

Não existe uma regra para determinar o tamanho de uma partição para o diretório `/home` (a partição inicial). É preciso levar em consideração o número de usuários no sistema e como ele será usado. Um usuário que apenas navega na web e processa textos requer menos espaço do que um que trabalha com edição de vídeo, por exemplo.

Dados variáveis (`/var`)

Este diretório contém “dados variáveis”, ou arquivos e diretórios nos quais o sistema deve poder escrever durante a operação. Dentre eles estão os logs do sistema (em `/var/log`), os arquivos temporários (`/var/tmp`) e os dados de aplicativos em cache (em `/var/cache`).

`/var/www/html` também é o diretório padrão para os arquivos de dados do Apache Web Server e `/var/lib/mysql` é o local padrão para os arquivos de banco de dados do servidor MySQL. Porém, ambos podem ser alterados.

Uma boa razão para colocar `/var` em uma partição separada é a estabilidade. Muitos aplicativos e processos escrevem em `/var` e seus subdiretórios, como `/var/log` ou `/var/tmp`. Um processo defeituoso poderia escrever dados até que não restasse nenhum espaço livre no sistema de arquivos.

Se `/var` estiver em `/`, isso poderia disparar um pânico do kernel e corromper o sistema de arquivos, causando uma situação muito difícil de revertir. Mas se `/var` for mantido em uma partição separada, o sistema de arquivos raiz não será afetado.

Como no caso de `/home`, não existe uma regra universal para determinar o tamanho da partição de `/var`, já que isso vai variar conforme o uso que for feito do sistema. Em um sistema doméstico, uns poucos gigabytes devem bastar. Mas em um banco de dados ou servidor web, será necessário bem mais espaço. Nesse caso, seria melhor colocar `/var` em uma partição de um disco diferente da partição raiz, adicionando uma camada extra de proteção contra falhas físicas do disco.

Swap

A partição de troca (ou swap) é usada para passar as páginas de memória da RAM para o disco conforme necessário. Esta partição precisa ser de um tipo específico e configurada com um utilitário apropriado chamado `mkswap` antes de poder ser usada.

A partição de troca não pode ser montada como as outras, o que significa que não é possível acessá-la como um diretório normal e visualizar seu conteúdo.

Um sistema pode ter diversas partições de troca (embora isso seja incomum), e o Linux também

suporta o uso de *arquivos* de troca em vez de partições, o que pode ser útil para aumentar rapidamente o espaço de troca quando necessário.

O tamanho da partição de troca é um tema controverso. A regra antiga dos primeiros dias do Linux (“duas vezes a quantidade de RAM”) nem sempre se aplica, dependendo de como o sistema está sendo usado e da quantidade de RAM física instalada.

Na documentação do Red Hat Enterprise Linux 7, a Red Hat recomenda o seguinte:

Quantidade de RAM	Tamanho de troca recomendado	Tamanho de troca recomendado com hibernação
< 2 GB de RAM	2x a quantidade de RAM	3x a quantidade de RAM
2-8 GB de RAM	Igual à quantidade de RAM	2x a quantidade de RAM
8-64 GB de RAM	No mínimo 4 GB	1.5x a quantidade de RAM
> 64 GB de RAM	No mínimo 4 GB	Não recomendado

Obviamente, a quantidade de troca pode depender da carga de trabalho. Se a máquina estiver executando um serviço crítico, como um banco de dados, servidor Web ou SAP, é aconselhável verificar o valor recomendado na documentação desses serviços (ou consultar o fornecedor do software).

NOTE

Para saber mais sobre a criação e ativação de espaços e arquivos de troca, consulte o Objetivo 104.1 do LPIC-1.

LVM

Já discutimos como os discos são organizados em uma ou mais partições, com cada partição contendo um sistema de arquivos que descreve como os arquivos e os metadados associados são armazenados. Uma das desvantagens do particionamento é que o administrador do sistema deve decidir antecipadamente como o espaço em disco disponível em um dispositivo será distribuído. Isso pode ser problemático mais tarde, se uma partição exigir mais espaço do que o originalmente planejado. Claro que as partições podem ser redimensionadas, mas isso nem sempre é possível (se, por exemplo, não houver mais espaço livre no disco).

O *Gerenciamento de Volumes Lógicos* (LVM) é uma forma de virtualização do armazenamento que oferece aos administradores de sistema um método mais flexível do que o particionamento tradicional para gerenciar o espaço em disco. O objetivo do LVM é facilitar a gestão das necessidades de armazenamento de seus usuários finais. A unidade básica é o *Volume Físico* (PV), que é um dispositivo de bloco no sistema, como uma partição de disco ou uma matriz RAID.

Os PVs são agrupados em *Grupos de Volumes* (VG), que abstraem os dispositivos subjacentes e são vistos como um único dispositivo lógico, com a capacidade de armazenamento combinada dos PVs do componente.

Cada volume em um Grupo de Volumes é subdividido em partes de tamanho fixo chamadas *extensões*. As extensões em um PV são chamadas *Extensões Físicas* (PE), enquanto as do volume lógico são *Extensões Lógicas* (LE). De maneira geral, cada extensão lógica é mapeada para uma extensão física, mas isso pode mudar se forem usados recursos como o espelhamento de disco.

Os Grupos de Volumes podem ser subdivididos em Volumes Lógicos (LVs), com funcionalidade semelhante à das partições, mas com mais flexibilidade.

O tamanho de um Volume Lógico, conforme especificado durante a sua criação, é na verdade definido pelo tamanho das extensões físicas (por padrão, 4 MB) multiplicado pelo número de extensões no volume. Com isso, é fácil entender que para aumentar um Volume Lógico, por exemplo, basta adicionar mais extensões do pool disponível no Grupo de Volumes. Da mesma forma, as extensões podem ser removidas para reduzir o LV.

Após sua criação, um Volume Lógico é visto pelo sistema operacional como um dispositivo de bloco normal. Um dispositivo será criado em /dev, com o nome /dev/VGNAME/LVNAME, em que VGNAME é o nome do Grupo de Volumes e LVNAME o nome do Volume Lógico.

Esses dispositivos podem ser formatados com o sistema de arquivos desejado usando utilitários padrão (como o mkfs.ext4, por exemplo) e montados com os métodos usuais, seja manualmente, com o comando mount, ou automaticamente, adicionando-os ao arquivo /etc/fstab.

Exercícios Guiados

- Nos sistemas Linux, onde são armazenados os arquivos do gerenciador de inicialização GRUB?

- Onde deve estar a partição de inicialização para garantir que um PC seja capaz de carregar o kernel?

- Onde a partição EFI costuma ser montada?

- Normalmente, em qual diretório se monta manualmente um sistema de arquivos?

Exercícios Exploratórios

1. Qual a menor unidade dentro de um Grupo de Volumes?

2. Como se define o tamanho de um Volume Lógico?

3. Em um disco formatado com o esquema de particionamento MBR, qual a ID da Partição do Sistema EFI?

4. Além das partições de troca, como é possível aumentar rapidamente o espaço de troca em um sistema Linux?

Resumo

Nesta lição, falamos sobre particionamento e explicamos quais diretórios geralmente são mantidos em partições separadas e por quê. Além disso, demos uma visão geral do LVM (Gerenciamento Lógico de Volumes) e como ele pode oferecer uma maneira mais flexível de alocar os dados e o espaço em disco em comparação com o particionamento tradicional.

Os seguintes arquivos, termos e utilitários foram discutidos:

/

O sistema de arquivos raiz do Linux.

/var

O local padrão dos “dados variáveis”, ou seja, dados que podem encolher ou aumentar com o tempo.

/home

O diretório pai padrão para as pastas iniciais dos usuários regulares de um sistema.

/boot

O local padrão dos arquivos do gerenciador de inicialização, kernel do Linux e disco de RAM inicial.

Partição do Sistema EFI (ESP)

Usada pelos sistemas que têm UEFI implementada para o armazenamento dos arquivos de inicialização do sistema.

Espaço de troca

Usado para trocar páginas de memória do kernel quando a RAM está sendo muito solicitada.

Pontos de montagem

Locais em um diretório nos quais um dispositivo (como um disco rígido) será montado.

Partições

Divisões em um disco rígido.

Respostas aos Exercícios Guiados

1. Nos sistemas Linux, onde são armazenados os arquivos do gerenciador de inicialização GRUB?
Em /boot/grub.
2. Onde deve estar a partição de inicialização para garantir que um PC seja capaz de carregar o kernel?

Antes do cilindro 1024.

3. Onde a partição EFI costuma ser montada?

Em /boot/efi.

4. Normalmente, em qual diretório se monta manualmente um sistema de arquivos?

Em /mnt. Porém, isso não é obrigatório. Podemos montar uma partição no diretório que quisermos.

Respostas aos Exercícios Exploratórios

1. Qual a menor unidade dentro de um Grupo de Volumes?

Os Grupos de Volumes são subdivididos em extensões.

2. Como se define o tamanho de um Volume Lógico?

Pelo tamanho das extensões físicas multiplicado pelo número de extensões no volume.

3. Em um disco formatado com o esquema de particionamento MBR, qual a ID da Partição do Sistema EFI?

A ID é 0xEF.

4. Além das partições de troca, como é possível aumentar rapidamente o espaço de troca em um sistema Linux?

Podem ser usados arquivos de troca.



102.2 Instalar o gerenciador de inicialização

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 102.2

Peso

2

Áreas chave de conhecimento

- Fornecer locais de boot alternativos e backup das opções de boot.
- Instalar e configurar um gerenciador de inicialização como o GRUB Legacy.
- Realizar mudanças na configuração básica do GRUB 2.
- Interagir com o carregador de boot.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- menu.lst, grub.cfg and grub.conf
- grub-install
- grub-mkconfig
- MBR



Linux
Professional
Institute

102.2 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	102 Instalação do Linux e gerenciamento de pacotes
Objetivo:	102.2 Instalar um gerenciador de inicialização
Lição:	1 de 1

Introdução

Quando um computador é ligado, o primeiro software a ser executado é o gerenciador de inicialização. O único objetivo desse código é carregar o kernel de um sistema operacional e entregar o controle a ele. O kernel então carrega os drivers necessários, inicializa o hardware e, em seguida, carrega o restante do sistema operacional.

O GRUB é o gerenciador de inicialização usado na maioria das distribuições Linux. Ele pode carregar o kernel do Linux ou outros sistemas operacionais, como o Windows, além de lidar com diversas imagens e parâmetros do kernel como entradas de menu separadas. A seleção do kernel na inicialização é feita por meio de uma interface acionada por teclado. Existe também uma interface de linha de comando para editar as opções e parâmetros de inicialização.

A maioria das distribuições Linux instala e configura o GRUB (na verdade, o GRUB 2) automaticamente, para que um usuário comum não precise pensar nisso. No entanto, como administrador do sistema, é essencial saber controlar o processo de inicialização para ser possível recuperar o sistema no caso de uma falha na inicialização – por exemplo, se houver um erro na atualização do kernel.

Nesta lição, você aprenderá como instalar, configurar e interagir com o GRUB.

GRUB Legacy e GRUB 2

A versão original do GRUB (*Grand Unified Bootloader*), agora conhecida como *GRUB Legacy*, foi desenvolvida em 1995 como parte do projeto GNU Hurd e, mais tarde, adotada como o gerenciador de inicialização padrão de muitas distribuições Linux, substituindo alternativas anteriores, como o LILO.

O GRUB 2 é uma remodelagem completa do GRUB com o objetivo de deixá-lo mais limpo, seguro, robusto e poderoso. Dentre as muitas vantagens em relação ao GRUB Legacy estão um arquivo de configuração muito mais flexível (com mais comandos e instruções condicionais, semelhantes a uma linguagem de script), um design mais modular e melhor localização/internacionalização.

Também há suporte para temas e menus gráficos de inicialização com telas de abertura, a possibilidade de inicializar ISOs LiveCD diretamente a partir do disco rígido, melhor suporte para arquiteturas não-x86, suporte universal para UUIDs (facilitando a identificação de discos e partições) e muito mais.

O GRUB Legacy não está mais em desenvolvimento ativo (a última versão foi a 0.97, em 2005), e hoje a maioria das principais distribuições Linux vem com o GRUB 2 como gerenciador de inicialização padrão. No entanto, ainda podemos encontrar sistemas usando o GRUB Legacy; portanto, é importante saber como usá-lo e em que ele difere do GRUB 2.

Onde fica o gerenciador de inicialização?

Historicamente, os discos rígidos nos sistemas compatíveis com IBM PC eram particionados usando o esquema de particionamento MBR, criado em 1982 para o IBM PC-DOS (MS-DOS) 2.0.

Nesse esquema, o primeiro setor de 512 bytes do disco é chamado de *Master Boot Record* (Registro Mestre de Inicialização, em português) e contém uma tabela que descreve as partições no disco (a tabela de partições) e também um código de inicialização (bootstrap), chamado gerenciador de inicialização.

Quando o computador é ligado, esse código mínimo (devido às restrições de tamanho) do gerenciador de inicialização é carregado, executado e transfere o controle para um carregador de inicialização secundário no disco, geralmente localizado em um espaço de 32 KB entre o MBR e a primeira partição, que por sua vez carregará o(s) sistema(s) operacional(is).

Em um disco particionado em MBR, o código de inicialização do GRUB é instalado no MBR. Ele carrega e transfere o controle para uma imagem “núcleo” instalada entre o MBR e a primeira partição. A partir desse ponto, o GRUB é capaz de carregar o restante dos recursos necessários

(definições de menu, arquivos de configuração e módulos extras) do disco.

No entanto, o MBR tem limitações no número de partições (originalmente, no máximo 4 partições primárias; mais tarde, no máximo 3 partições primárias com 1 partição estendida, subdividida em uma série de partições lógicas) e no tamanho máximo de disco (2 TB). Para suplantar essas limitações, foi criado um novo esquema de particionamento chamado GPT (*GUID Partition Table*), parte do padrão UEFI (*Unified Extensible Firmware Interface*).

Os discos particionados em GPT podem ser usados em computadores com a BIOS tradicional do PC ou com o firmware UEFI. Em máquinas com BIOS, a segunda parte do GRUB é armazenada em uma partição especial de inicialização da BIOS.

Em sistemas com firmware UEFI, o GRUB é carregado pelo firmware a partir dos arquivos `grubia32.efi` (para sistemas de 32 bits) ou `grubx64.efi` (para sistemas de 64 bits) em uma partição chamada ESP (*EFI System Partition*)

A partição /boot

No Linux, os arquivos necessários para o processo de inicialização geralmente são armazenados em uma partição de inicialização montada no sistema de arquivos raiz e coloquialmente chamada de `/boot`.

Uma partição de inicialização não é imprescindível nos sistemas atuais, já que os carregadores de inicialização como o GRUB geralmente são capazes de montar o sistema de arquivos raiz e procurar os arquivos necessários dentro de um diretório `/boot`. Porém, seu uso é aconselhável, pois ela separa os arquivos necessários ao processo de inicialização do restante do sistema de arquivos.

Essa partição geralmente é a primeira do disco. A razão para isso é que a IBM PC BIOS original endereçava os discos usando *cilindros, cabeças e setores* (CHS), com um máximo de 1024 cilindros, 256 cabeças e 63 setores, resultando em um tamanho máximo de disco de 528 MB (504 MB no MS-DOS). Isso significa que qualquer coisa além dessa marca não estaria acessível em sistemas legados, a menos que um esquema de endereçamento de disco diferente (como o *Endereçamento de bloco lógico*, LBA) fosse usado.

Portanto, para garantir uma compatibilidade máxima, a partição `/boot` geralmente está localizada no início do disco e termina antes do cilindro 1024 (528 MB), garantindo que a máquina sempre possa carregar o kernel. O tamanho recomendado para essa partição em uma máquina atual é de 300 MB.

Outras razões para uma partição `/boot` separada são a criptografia e a compactação, já que alguns métodos ainda não são suportados pelo GRUB 2, ou ainda se a partição raiz do sistema (/) precisar ser formatada usando um sistema de arquivos não suportado.

Conteúdo da partição de inicialização

O conteúdo da partição `/boot` pode variar de acordo com a arquitetura do sistema ou o gerenciador de inicialização usado, mas em um sistema baseado em x86 geralmente encontramos os arquivos abaixo. A maioria deles recebe o sufixo `-VERSION`, onde `-VERSION` é a versão correspondente do kernel do Linux. Assim, por exemplo, o arquivo de configuração para a versão do kernel do Linux `4.15.0-65-generic` seria chamado de `config-4.15.0-65-generic`.

Arquivo de configuração

Este arquivo, geralmente chamado `config-VERSION` (veja o exemplo acima), armazena parâmetros de configuração para o kernel do Linux. Este arquivo é gerado automaticamente quando um novo kernel é compilado ou instalado e *não deve ser diretamente modificado pelo usuário*.

Mapa do sistema

Este arquivo é uma tabela de consulta que articula nomes de símbolos (como variáveis ou funções) com sua posição correspondente na memória. Isso é útil ao se corrigir um tipo de falha do sistema conhecida como *kernel panic* (pânico do kernel), pois permite ao usuário saber qual variável ou função estava sendo chamada quando ocorreu a falha. Como no caso do arquivo de configuração, o nome geralmente é `System.map-VERSION` (por exemplo, `System.map-4.15.0-65-generic`).

Kernel do Linux

Este é o kernel do sistema operacional propriamente dito. Seu nome é geralmente `vmlinuz-VERSION` (por exemplo, `vmlinuz-4.15.0-65-generic`). Também se pode encontrar o nome `vmlinuz` em vez de `vmlinuz` – o `z` no final indica que o arquivo foi compactado.

Disco RAM inicial

Geralmente se chama `initrd.img-VERSION` e contém um sistema de arquivos raiz mínimo carregado em um disco RAM, contendo os utilitários e módulos de kernel necessários para que o kernel possa montar o sistema de arquivos raiz real.

Arquivos relacionados ao gerenciador de inicialização

Em sistemas com o GRUB instalado, eles costumam estar localizados em `/boot/grub` e incluem o arquivo de configuração (`/boot/grub/grub.cfg` para o GRUB 2 ou `/boot/grub/menu.lst` no caso do GRUB Legacy), módulos (em `/boot/grub/i386-pc`), arquivos de tradução (em `/boot/grub/locale`) e fontes (em `/boot/grub/fonts`) do GRUB.

GRUB 2

Instalando o GRUB 2

O GRUB 2 pode ser instalado usando o utilitário `grub-install`. Se seu sistema se recusa a inicializar, você precisará inicializá-lo usando um Live CD ou um disco de recuperação, descobrir qual é a partição de inicialização do seu sistema, montá-la e executar o utilitário.

NOTE

Para usar os comandos abaixo, é preciso estar logado como root. Se não for o caso, execute primeiro `sudo su` – para “virar” root. No final, digite `exit` para se deslogar e voltar a ser um usuário comum.

O primeiro disco de um sistema é geralmente o *dispositivo de inicialização*. Pode ser necessário saber se existe uma *partição de inicialização* no disco. Para isso existe o utilitário `fdisk`. Para listar todas as partições no primeiro disco da sua máquina, use:

```
# fdisk -l /dev/sda
Disk /dev/sda: 111,8 GiB, 120034123776 bytes, 234441648 sectors
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    *       2048  2000895  1998848  976M 83 Linux
/dev/sda2          2002942 234440703 232437762 110,9G  5 Extended
/dev/sda5          2002944 18008063 16005120   7,6G 82 Linux swap / Solaris
/dev/sda6          18010112 234440703 216430592 103,2G 83 Linux
```

A partição de inicialização é identificada com o * na coluna boot. No exemplo acima, ela é `/dev/sda1`.

A seguir, crie um diretório temporário em `/mnt` e monte a partição nele:

```
# mkdir /mnt/tmp
# mount /dev/sda1 /mnt/tmp
```

Depois execute `grub-install`, apontando-o para o *dispositivo de inicialização* (*não* a partição) e o diretório em que a partição de inicialização está montada. Se o sistema possuir uma partição de inicialização dedicada, o comando será:

```
# grub-install --boot-directory=/mnt/tmp /dev/sda
```

Caso esteja instalando em um sistema que não possui uma partição de inicialização, mas apenas um diretório /boot no sistema de arquivos raiz, aponte-o em grub-install. Nesse caso, o comando é:

```
# grub-install --boot-directory=/boot /dev/sda
```

Configurando o GRUB 2

O arquivo de configuração padrão do GRUB 2 é /boot/grub/grub.cfg. Esse arquivo é gerado automaticamente e não se recomenda editá-lo à mão. Para fazer alterações na configuração do GRUB, é preciso editar o arquivo /etc/default/grub e depois executar o utilitário update-grub para gerar um arquivo compatível.

NOTE

update-grub normalmente é um atalho para grub-mkconfig -o /boot/grub/grub.cfg, de modo que eles produzem os mesmos resultados.

Existem algumas opções no arquivo /etc/default/grub para controlar o comportamento do GRUB 2, como o kernel padrão de inicialização, tempo limite, parâmetros extras da linha de comando, etc. Os mais importantes são:

GRUB_DEFAULT=

A entrada de menu padrão para a inicialização. Pode ser um valor numérico (como 0, 1, etc.), o nome de um item de menu (como debian) ou saved, que é usado em conjunto com GRUB_SAVEDefault=, explicado abaixo. Lembre-se de que as entradas de menu começam com zero, de forma que a primeira é 0, a segunda é 1 e assim por diante.

GRUB_SAVEDefault=

Se esta opção estiver definida como true e GRUB_DEFAULT= como saved, a opção padrão de inicialização sempre será a última selecionada no menu de inicialização.

GRUB_TIMEOUT=

O tempo limite, em segundos, para que a entrada do menu padrão seja selecionada. Se definido como 0, o sistema inicializará a entrada padrão sem exibir um menu. Se definido como -1, o sistema aguardará até que o usuário selecione uma opção, não importa quanto tempo leve.

GRUB_CMDLINE_LINUX=

Lista as opções de linha de comando que serão adicionadas às entradas do kernel do Linux.

GRUB_CMDLINE_LINUX_DEFAULT=

Por padrão, duas entradas de menu são geradas para cada kernel do Linux, uma com as opções padrão e uma entrada para recuperação. Com esta opção, você pode incluir parâmetros extras que

serão adicionados apenas à entrada padrão.

GRUB_ENABLE_CRYPTODISK=

Se definido como `y`, comandos como `grub-mkconfig`, `update-grub` e `grub-install` procuram por discos criptografados e adicionam os comandos necessários para acessá-los durante a inicialização. Assim, a inicialização automática é desabilitada (`GRUB_TIMEOUT=` com valor diferente de `-1`) porque uma senha será exigida para descriptografar os discos antes que possam ser acessados.

Administrando as entradas de menu

Quando rodamos o `update-grub`, o GRUB 2 busca por kernels e sistemas operacionais na máquina e gera as entradas de menu correspondentes no arquivo `/boot/grub/grub.cfg`. É possível adicionar novas entradas manualmente aos arquivos de script dentro do diretório `/etc/grub.d`.

Esses arquivos devem ser executáveis e são processados em ordem numérica pelo `update-grub`. Portanto, `05_debian_theme` é processado antes de `10_linux` e assim por diante. As entradas de menu personalizadas são geralmente adicionadas ao arquivo `40_custom`.

A sintaxe básica de uma entrada de menu é mostrada abaixo:

```
menuentry "Default OS" {  
    set root=(hd0,1)  
    linux /vmlinuz root=/dev/sda1 ro quiet splash  
    initrd /initrd.img  
}
```

A primeira linha sempre começa com `menuentry` e termina com `{`. O texto entre aspas será mostrado como o rótulo da entrada no menu de inicialização do GRUB 2.

O parâmetro `set root` define o disco e a partição em que o sistema de arquivos raiz do sistema operacional está localizado. Note que no GRUB 2 os discos são numerados a partir de zero, então `hd0` é o primeiro disco (`sda` no Linux), `hd1` o segundo e assim por diante. As partições, por sua vez, são numeradas a partir de um. No exemplo acima, o sistema de arquivos raiz está localizado no primeiro disco (`hd0`), primeira partição (`,1`) ou `sda1`.

Em vez de especificar diretamente o dispositivo e a partição, também podemos pedir que o GRUB 2 busque por um sistema de arquivos com um rótulo ou UUID (*Universally Unique Identifier*) específico. Para isso, use o parâmetro `search --set=root` seguido por `--label` e o rótulo do sistema de arquivos a buscar, ou `--fs-uuid` seguido pelo UUID do sistema de arquivos.

Use o comando abaixo para encontrar o UUID de um sistema de arquivos:

```
$ ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx 1 root root 10 nov  4 08:40 3e0b34e2-949c-43f2-90b0-25454ac1595d ->
../../sda5
lrwxrwxrwx 1 root root 10 nov  4 08:40 428e35ee-5ad5-4dcb-adca-539aba6c2d84 ->
../../sda6
lrwxrwxrwx 1 root root 10 nov  5 19:10 56C11DCC5D2E1334 -> ../../sdb1
lrwxrwxrwx 1 root root 10 nov  4 08:40 ae71b214-0aec-48e8-80b2-090b6986b625 ->
../../sda1
```

No exemplo acima, o UUID de `/dev/sda1` é `ae71b214-0aec-48e8-80b2-090b6986b625`. Se quisermos defini-lo como dispositivo raiz do GRUB 2, o comando seria `search --set=root --fs-uuid ae71b214-0aec-48e8-80b2-090b6986b625`.

Ao usar o comando `search`, é comum adicionar o parâmetro `--no-floppy` para que o GRUB não perca tempo buscando em disquetes.

A linha `linux` indica onde está localizado o kernel do sistema operacional (neste caso, o arquivo `vmlinuz` na raiz do sistema de arquivos). Depois disso, podemos passar parâmetros na linha de comando para o kernel.

No exemplo acima, especificamos a partição root (`root=/dev/sda1`) e passamos três parâmetros do kernel: a partição root deve ser montada com acesso apenas de leitura (`r0`), a maioria das mensagens de log deve ser desabilitada (`quiet`) e uma tela de boas-vindas deve ser exibida (`splash`).

A linha `initrd` indica onde está localizado o disco de RAM inicial. No exemplo acima, o arquivo é `initrd.img`, localizado na raiz do sistema de arquivos.

NOTE

A maioria das distribuições Linux não coloca de fato o kernel e o initrd no diretório raiz do sistema de arquivos raiz. Em vez disso, esses são links para os arquivos reais dentro do diretório ou partição `/boot`.

A última linha de uma entrada do menu deve conter apenas o caractere `}`.

Interagindo com o GRUB 2

Ao inicializar um sistema com o GRUB 2 aparece um menu de opções. Use as setas do teclado para selecionar uma opção e `Enter` para confirmar e inicializar a entrada selecionada.

TIP

Se você vir apenas uma contagem regressiva, mas não um menu, pressione `Shift` para

exibir o menu.

Para editar uma opção, selecione-a com as setas e pressione `E`. Será exibida uma janela de editor com o conteúdo da `menuentry` associada àquela opção, confirme definido em `/boot/grub/grub.cfg`.

Depois de editar uma opção, digite `Ctrl + X` ou `F10` para inicializar, ou ainda `Esc` para retornar ao menu.

Para entrar no shell do GRUB 2, pressione `C` na tela do menu (ou `Ctrl + C`) na janela de edição. Aparecerá um prompt de comando como este: `grub >`

Digite `help` para ver uma lista de todos os comandos disponíveis ou pressione `Esc` para sair do shell e retornar à tela do menu.

NOTE

Lembre-se de que esse menu não aparecerá se `GRUB_TIMEOUT` estiver definido com `0` em `/etc/default/grub`.

Inicialização a partir do shell do GRUB 2

Podemos usar o shell do GRUB 2 para inicializar o sistema caso uma configuração incorreta em uma entrada de menu cause uma falha de inicialização.

A primeira coisa que você deve fazer é descobrir onde está a partição de inicialização. Use para isso o comando `ls`, que mostra uma lista das partições e discos encontrados pelo GRUB 2.

```
grub> ls
(proc) (hd0) (hd0,msdos1)
```

No exemplo acima, tudo é simples. Existe apenas um disco (`hd0`), com apenas uma partição: (`hd0,msdos1`).

Os discos e partições listados serão diferentes no seu sistema. Em nosso exemplo, a primeira partição do `hd0` é chamada `msdos1` porque o disco foi particionado usando o esquema de particionamento MBR. Se ele fosse particionado usando GPT, o nome seria `gpt1`.

Para inicializar o Linux, precisamos de um kernel e de um disco RAM inicial (`initrd`). Vamos verificar o conteúdo de (`hd0,msdos1`):

```
grub> ls (hd0,msdos1)/
lost+found/ swapfile etc/ media/ bin/ boot/ dev/ home/ lib/ lib64/ mnt/ opt/ proc/
```

```
root/ run/ sbin/ srv/ sys/ tmp/ usr/ var/ initrd.img initrd.img.old vmlinuz cdrom/
```

Podemos adicionar o parâmetro `-l` a `ls` para obter uma lista longa, como faríamos em um terminal Linux. Use `Tab` para completar automaticamente os nomes de discos, partições e arquivos.

Note que temos as imagens do kernel (`vmlinuz`) e `initrd` (`initrd.img`) bem no diretório raiz. Se não, poderíamos conferir o conteúdo de `/boot` com `list (hd0,msdos1)/boot/`.

Agora, defina a partição de inicialização:

```
grub> set root=(hd0,msdos1)
```

Carregue o kernel do Linux com o comando `linux`, seguido pelo caminho para o kernel e pela opção `root=` para informar ao kernel onde está localizado o sistema de arquivos raiz do sistema operacional.

```
grub> linux /vmlinuz root=/dev/sda1
```

Carregue o disco RAM inicial com `initrd`, seguido pelo caminho completo para o arquivo `initrd.img`:

```
grub> initrd /initrd.img
```

Agora, initialize o sistema com `boot`.

Inicializando com o shell de resgate

No caso de uma falha na inicialização, o GRUB 2 pode carregar um shell de resgate, uma versão simplificada do shell que mencionamos anteriormente. Você o reconhecerá pelo prompt de comando, que é exibido como `grub rescue>`.

O processo para inicializar um sistema a partir deste shell é quase idêntico ao mostrado anteriormente. No entanto, será preciso carregar alguns módulos do GRUB 2 para fazer as coisas funcionarem.

Depois de descobrir qual é a partição de inicialização (com `ls`, como mostrado anteriormente), use o comando `set prefix=`, seguido pelo caminho completo para o diretório que contém os arquivos do GRUB 2 – geralmente `/boot/grub`. No nosso exemplo:

```
grub rescue> set prefix=(hd0,msdos1)/boot/grub
```

Em seguida, carregue os módulos `normal` e `linux` com o comando `insmod`:

```
grub rescue> insmod normal  
grub rescue> insmod linux
```

A seguir, defina a partição de inicialização com `set root=` como ensinado anteriormente, carregue o kernel do `linux` (com `linux`), o disco RAM inicial (`initrd`) e tente inicializar com `boot`.

GRUB Legacy

Instalando o GRUB Legacy a partir de um sistema em execução

Para instalar o GRUB Legacy em um disco a partir de um sistema em execução, empregaremos o utilitário `grub-install`. O comando básico é `grub-install DEVICE`, onde `DEVICE` é o disco no qual você deseja instalar o GRUB Legacy. Um exemplo seria `/dev/sda`.

```
# grub-install /dev/sda
```

Note que é preciso especificar o *dispositivo* no qual o GRUB Legacy será instalado, como `/dev/sda/`, *não a partição* como em `/dev/sda1`.

Por padrão, o GRUB copia os arquivos necessários para o diretório `/boot` no dispositivo especificado. Se você deseja copiá-los para outro diretório, use o parâmetro `--boot-directory=` seguido pelo caminho completo para o local onde os arquivos devem ser copiados.

Instalando o GRUB Legacy a partir de um shell do GRUB

Se você não conseguir inicializar o sistema por algum motivo e precisar reinstalar o GRUB Legacy, poderá fazê-lo no shell do GRUB em um disco de inicialização do GRUB Legacy.

No shell do GRUB (digite `c` no menu de inicialização para acessar o prompt `grub>`), o primeiro passo é configurar o dispositivo de inicialização, que contém o diretório `/boot`. Por exemplo, se esse diretório estiver na primeira partição do primeiro disco, o comando seria:

```
grub> root (hd0,0)
```

Se você não souber qual dispositivo contém o diretório `/boot`, peça ao GRUB para procurá-lo com o comando `find`, como abaixo:

```
grub> find /boot/grub/stage1  
(hd0,0)
```

Em seguida, defina a partição de inicialização conforme as instruções acima e use o comando `setup` para instalar o GRUB Legacy no MBR e copiar os arquivos necessários no disco:

```
grub> setup (hd0)
```

Ao final, reinicie o sistema e ele deverá inicializar normalmente.

Definindo entradas e configurações do menu do GRUB Legacy

As entradas de menu e configurações do GRUB Legacy são armazenadas no arquivo `/boot/grub/menu.lst`. Trata-se de uma lista de comandos e parâmetros em um arquivo de texto simples, que pode ser editado diretamente em seu editor de texto predileto.

As linhas que começam com `#` são consideradas comentários e as linhas em branco são ignoradas.

Uma entrada de menu possui ao menos três comandos. O primeiro, `title`, define o título do sistema operacional na tela do menu. O segundo, `root`, informa ao GRUB Legacy qual o dispositivo ou partição de inicialização.

A terceira entrada, `kernel`, especifica o caminho completo para a imagem do kernel que deve ser carregada quando a entrada correspondente for selecionada. Observe que esse caminho é relativo ao dispositivo especificado no parâmetro `root`.

Veja um exemplo simples a seguir:

```
# This line is a comment  
title My Linux Distribution  
root (hd0,0)  
kernel /vmlinuz root=/dev/hda1
```

Ao contrário do GRUB 2, no GRUB Legacy tanto as partições quanto os discos são numerados a partir de zero. Portanto, o comando `root (hd0,0)` define a partição de inicialização como a primeira partição (`0`) do primeiro disco (`hd0`).

Podemos omitir a instrução `root` se especificarmos o dispositivo de inicialização antes do caminho no comando `kernel`. A sintaxe é a mesma, portanto:

```
kernel (hd0,0)/vmlinuz root=/dev/hda1
```

equivale a:

```
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
```

Ambos carregam o arquivo `vmlinuz` a partir do diretório `root (/)` da primeira partição do primeiro disco (`hd0,0`).

O parâmetro `root=/dev/hda1` após o comando `kernel` informa ao kernel do Linux qual partição deve ser usada como sistema de arquivos raiz. Este é um parâmetro do kernel do Linux, não um comando do GRUB Legacy.

NOTE

Para saber mais sobre os parâmetros do kernel, visite <https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html>.

Pode ser necessário especificar o local da imagem inicial do disco RAM para o sistema operacional com o parâmetro `initrd`. O caminho completo para o arquivo pode ser especificado como no parâmetro `kernel`, mas também é possível especificar um dispositivo ou partição antes do caminho, como por exemplo:

```
# This line is a comment
title My Linux Distribution
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
initrd /initrd.img
```

O GRUB Legacy tem um design modular, no qual módulos (geralmente armazenados como arquivos `.mod` em `/boot/grub/i386-pc`) podem ser carregados para adicionar recursos extras, como suporte a hardware incomum, sistemas de arquivos ou novos algoritmos de compactação.

Os módulos são carregados usando o comando `module`, seguido pelo caminho completo para o arquivo `.mod` correspondente. Lembre-se de que, como no caso dos kernels e imagens `initrd`, esse caminho é relativo ao dispositivo especificado no comando `root`.

O exemplo abaixo carrega o módulo `915resolution`, necessário para definir corretamente a resolução da memória de imagens (frame buffer) em sistemas com chipsets de vídeo das séries Intel 800 ou 900.

```
module /boot/grub/i386-pc/915resolution.mod
```

Carregamento em cadeia de outros sistemas operacionais

O GRUB Legacy pode ser usado para carregar sistemas operacionais não suportados, como o Windows, usando um processo chamado de *carregamento em cadeia* (chainloading). O GRUB Legacy é carregado primeiro e, quando a opção correspondente é selecionada, o gerenciador de inicialização do sistema desejado é carregado.

Uma entrada típica para o carregamento em cadeia do Windows seria semelhante a esta:

```
# Load Windows
title Windows XP
root (hd0,1)
makeactive
chainload +1
boot
```

Vamos esclarecer cada parâmetro. Como anteriormente, `root (hd0,1)` especifica o dispositivo e a partição em que o gerenciador de inicialização do sistema operacional que queremos carregar está localizado. Neste exemplo, a *segunda* partição do primeiro disco.

makeactive

define um sinalizador indicando que esta é uma partição ativa. Funciona apenas em partições primárias do DOS.

chainload +1

diz ao GRUB para carregar o primeiro setor da partição de inicialização. É nela que costumam ficar os gerenciadores de inicialização.

boot

executa o gerenciador de inicialização e carrega o sistema operacional correspondente.

Exercícios Guiados

1. Qual é o local padrão do arquivo de configuração do GRUB 2?

2. Quais são as etapas necessárias para alterar as configurações do GRUB 2?

3. Em qual arquivo devem ser adicionadas as entradas de menu personalizadas do GRUB 2?

4. Onde são armazenadas as entradas de menu do GRUB Legacy?

5. Como podemos entrar no shell do GRUB a partir de um menu do GRUB 2 ou GRUB Legacy?

Exercícios Exploratórios

1. Imagine um usuário configurando o GRUB Legacy para inicializar a partir da segunda partição do primeiro disco. Ele escreve a seguinte entrada de menu personalizada:

```
title My Linux Distro
root (hd0,2)
kernel /vmlinuz root=/dev/hda1
initrd /initrd.img
```

No entanto, o sistema não inicializa. O que está errado?

2. Imagine que você tenha um disco identificado como `/dev/sda` com diversas partições. Que comando pode ser usado para descobrir qual a partição de inicialização em um sistema?

3. Qual o comando para descobrir o UUID de uma partição?

4. Considere a seguinte entrada para o GRUB 2

```
menuentry "Default OS" {
    set root=(hd0,1)
    linux /vmlinuz root=/dev/sda1 ro quiet splash
    initrd /initrd.img
}
```

Altere-a para que o sistema inicialize a partir de um disco com o UUID `5dda0af3-c995-481a-a6f3-46dc3b6998d`

5. Como configurar o GRUB 2 para aguardar 10 segundos antes de iniciar a entrada de menu padrão?

6. Em um shell do GRUB Legacy, quais são os comandos para instalar o GRUB na primeira partição do segundo disco?



Resumo

Nesta lição aprendemos

- O que é um carregador de inicialização.
- As diferenças entre o GRUB Legacy e o GRUB 2.
- O que é uma partição de inicialização e qual seu conteúdo.
- Como instalar o GRUB Legacy e o GRUB 2.
- Como configurar o GRUB Legacy e o GRUB 2.
- Como adicionar entradas de menu personalizadas ao GRUB Legacy e ao GRUB 2.
- Como interagir com a tela de menu e o console do GRUB Legacy e do GRUB 2.
- Como inicializar um sistema a partir de um shell do GRUB Legacy, do GRUB 2 ou do shell de resgate.

Os seguintes comandos foram abordados nesta lição:

- `grub-install`
- `update-grub`
- `grub-mkconfig`

Respostas aos Exercícios Guiados

1. Qual é o local padrão do arquivo de configuração do GRUB 2?

/boot/grub/grub.cfg

2. Quais são as etapas necessárias para alterar as configurações do GRUB 2?

Efetuar as alterações no arquivo /etc/default/grub, depois atualizar a configuração com update-grub.

3. Em qual arquivo devem ser adicionadas as entradas de menu personalizadas do GRUB 2?

/etc/grub.d/40_custom

4. Onde são armazenadas as entradas de menu do GRUB Legacy?

/boot/grub/menu.lst

5. Como podemos entrar no shell do GRUB a partir de um menu do GRUB 2 ou GRUB Legacy?

Pressionando c na tela de menu.

Respostas aos Exercícios Exploratórios

- Imagine um usuário configurando o GRUB Legacy para inicializar a partir da segunda partição do primeiro disco. Ele escreve a seguinte entrada de menu personalizada:

```
title My Linux Distro
root (hd0,2)
kernel /vmlinuz root=/dev/hda1
initrd /initrd.img
```

No entanto, o sistema não inicializa. O que está errado?

A partição de inicialização está incorreta. Lembre-se de que, ao contrário do GRUB 2, o GRUB Legacy conta as partições a partir de *zero*. Portanto, o comando correto para a segunda partição do primeiro disco seria `root (hd0,1)`.

- Imagine que você tenha um disco identificado como `/dev/sda` com diversas partições. Que comando pode ser usado para descobrir qual a partição de inicialização em um sistema?

Use `fdisk -l /dev/sda`. A partição de inicialização estará marcada na lista com um asterisco (*).

- Qual o comando para descobrir o UUID de uma partição?

Use `ls -la /dev/disk/by-uuid/` e procure pelo UUID que aponta para a partição.

- Considere a seguinte entrada para o GRUB 2

```
menuentry "Default OS" {
    set root=(hd0,1)
    linux /vmlinuz root=/dev/sda1 ro quiet splash
    initrd /initrd.img
}
```

Altere-a para que o sistema inicialize a partir de um disco com o UUID `5dda0af3-c995-481a-a6f3-46dc3b6998d`

Será preciso alterar a declaração `set root`. Em vez de especificar um disco e uma partição, diga ao GRUB para buscar pela partição com o UUID desejado.

```
menuentry "Default OS" {
    search --set=root --fs-uuid 5dda0af3-c995-481a-a6f3-46dc3b6998d
```

```
linux /vmlinuz root=/dev/sda1 ro quiet splash
initrd /initrd.img
}
```

5. Como configurar o GRUB 2 para aguardar 10 segundos antes de inicializar a entrada de menu padrão?

Adicionando o parâmetro GRUB_TIMEOUT=10 a /etc/default/grub.

6. Em um shell do GRUB Legacy, quais são os comandos para instalar o GRUB na primeira partição do segundo disco?

```
grub> root (hd1,0)
grub> setup (hd1)
```



102.3 Controle de bibliotecas compartilhadas

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 102.3

Peso

1

Áreas chave de conhecimento

- Identificar as bibliotecas compartilhadas.
- Identificar onde geralmente essas bibliotecas se localizam no sistema.
- Carregar as bibliotecas compartilhadas.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `ldd`
- `ldconfig`
- `/etc/ld.so.conf`
- `LD_LIBRARY_PATH`



102.3 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	102 Instalação do Linux e gerenciamento de pacotes
Objetivo:	102.3 Gerenciar bibliotecas compartilhadas
Lição:	1 de 1

Introdução

Nesta lição, trataremos das *bibliotecas compartilhadas*, também conhecidas como *objetos compartilhados*: trechos de código compilado e reutilizável, como funções ou classes, usados de forma recorrente por diversos programas.

Para começar, explicaremos o que são bibliotecas compartilhadas, como identificá-las e onde são encontradas. A seguir, veremos como configurar seus locais de armazenamento. Por fim, mostraremos como procurar pelas bibliotecas compartilhadas das quais depende um determinado programa.

O que são bibliotecas compartilhadas

As bibliotecas de software são coleções de código que podem ser usadas por vários programas diferentes, assim como as bibliotecas físicas permitem que livros e outros recursos sejam usados por várias pessoas diferentes.

Há duas etapas importantes na criação de um arquivo executável a partir do código fonte de um programa. Primeiro, o *compilador* transforma o código fonte em código de máquina, que é

armazenado nos chamados *arquivos-objeto*. Em segundo lugar, o *vinculador* (linker) combina os arquivos-objeto e os *vincula* às bibliotecas para gerar o arquivo executável final. Essa ligação pode ser feita de maneira *estática* ou *dinâmica*. Dependendo do método escolhido, falaremos em bibliotecas estáticas ou, no caso de um vínculo dinâmico, em bibliotecas compartilhadas. Vamos explicar as diferenças entre elas.

Bibliotecas estáticas

Uma biblioteca estática é mesclada com o programa no momento do vínculo. Uma cópia do código da biblioteca é incorporada ao programa e se torna parte dele. Portanto, o programa não tem dependências na biblioteca em tempo de execução, pois já contém o código da biblioteca. Não ter dependências pode ser visto como uma vantagem, pois não precisamos nos preocupar em garantir que as bibliotecas usadas estejam sempre disponíveis. Pelo lado negativo, os programas vinculados estaticamente são mais pesados.

Bibliotecas compartilhadas (ou dinâmicas)

No caso de bibliotecas compartilhadas, o vinculador simplesmente cuida para que o programa faça referência às bibliotecas corretamente. No entanto, o vinculador não copia nenhum código da biblioteca para o arquivo do programa. Assim, a biblioteca compartilhada deve estar disponível em tempo de execução para satisfazer as dependências do programa. Essa é uma abordagem econômica para gerenciar recursos do sistema, pois ajuda a reduzir o tamanho dos arquivos de programas; apenas uma cópia da biblioteca é carregada na memória, mesmo quando ela é usada por vários programas.

Convenções de nomenclatura para arquivos-objeto compartilhados

O nome de uma biblioteca compartilhada, também chamado de *soname*, segue um padrão composto por três elementos:

- Nome da biblioteca (normalmente com o prefixo `lib`)
- `so` (que significa “shared object”, ou objeto compartilhado)
- Número de versão da biblioteca

Eis um exemplo: `libpthread.so.0`

Em comparação, os nomes das bibliotecas estáticas terminam em `.a`, por exemplo `libpthread.a`.

NOTE

Como os arquivos que contêm bibliotecas compartilhadas precisam estar disponíveis quando o programa é executado, a maioria dos sistemas Linux já contém bibliotecas compartilhadas. Uma vez que as bibliotecas estáticas são necessárias apenas em um arquivo dedicado quando um programa é vinculado, elas

podem não estar presentes no sistema do usuário final.

A `glibc` (biblioteca GNU C) é um bom exemplo de biblioteca compartilhada. Em um sistema Debian GNU/Linux 9.9, seu arquivo se chama `libc.so.6`. Esses nomes de arquivo mais gerais normalmente são links simbólicos apontando para o arquivo real que contém uma biblioteca, cujo nome contém o número exato da versão. No caso da `glibc`, esse link simbólico é assim:

```
$ ls -l /lib/x86_64-linux-gnu/libc.so.6
lrwxrwxrwx 1 root root 12 feb  6 22:17 /lib/x86_64-linux-gnu/libc.so.6 -> libc-
2.24.so
```

Essa maneira de usar nomes de arquivos mais gerais para fazer referência a arquivos de bibliotecas compartilhadas nomeados por uma versão específica é uma prática comum.

Outros exemplos de bibliotecas compartilhadas incluem `libreadline` (que permite aos usuários editar linhas de comando à medida que são digitadas e inclui suporte para os modos de edição Emacs e vi), `libcrypt` (que contém funções relacionadas à criptografia, hash e codificação), ou `libcurl` (que é uma biblioteca multiprotocolo de transferência de arquivos). Estes são os locais comuns para bibliotecas compartilhadas em um sistema Linux:

- `/lib`
- `/lib32`
- `/lib64`
- `/usr/lib`
- `/usr/local/lib`

NOTE

O conceito de bibliotecas compartilhadas não é exclusivo do Linux. No Windows, por exemplo, elas são chamadas de DLL, que significa *bibliotecas de vínculo dinâmico*.

Configuração dos caminhos da biblioteca compartilhada

As referências contidas nos programas vinculados dinamicamente são resolvidas pelo vinculador dinâmico (`ld.so` ou `ld-linux.so`) quando o programa é executado. O vinculador dinâmico busca por bibliotecas em uma série de diretórios. Esses diretórios são especificados pelo *caminho da biblioteca*. O caminho da biblioteca é configurado no diretório `/etc`, especificamente no arquivo `/etc/ld.so.conf` e, o que atualmente é mais comum, nos arquivos do diretório `/etc/ld.so.conf.d`. Normalmente, o primeiro inclui uma única linha `include` para os arquivos `*.conf` do segundo:

```
$ cat /etc/ld.so.conf
include /etc/ld.so.conf.d/*.conf
```

O diretório `/etc/ld.so.conf.d` contém arquivos `*.conf`:

```
$ ls /etc/ld.so.conf.d/
libc.conf  x86_64-linux-gnu.conf
```

Esses arquivos `*.conf` devem incluir os caminhos absolutos para os diretórios da biblioteca compartilhada:

```
$ cat /etc/ld.so.conf.d/x86_64-linux-gnu.conf
# Multiarch support
/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu
```

O comando `ldconfig` trata de ler esses arquivos de configuração, criando o conjunto de links simbólicos anteriormente mencionados que ajudam a localizar as bibliotecas individuais, e, por fim, de atualizar o arquivo de cache `/etc/ld.so.cache`. Assim, o `ldconfig` deve ser executado sempre que atualizarmos ou adicionarmos arquivos de configuração.

Eis algumas opções úteis para o `ldconfig`:

-v, --verbose

Exibe os números da versão da biblioteca, o nome de cada diretório e os vínculos criados:

```
$ sudo ldconfig -v
/usr/local/lib:
/lib/x86_64-linux-gnu:
    libnss_myhostname.so.2 -> libnss_myhostname.so.2
    libfuse.so.2 -> libfuse.so.2.9.7
    libidn.so.11 -> libidn.so.11.6.16
    libnss_mdns4.so.2 -> libnss_mdns4.so.2
    libparted.so.2 -> libparted.so.2.0.1
    (...)
```

Assim, podemos ver, por exemplo, como `libfuse.so.2` está vinculado ao arquivo-objeto compartilhado real `libfuse.so.2.9.7`.

-p, --print-cache

Exibe as listas de diretórios e bibliotecas candidatas armazenadas na cache atual:

```
$ sudo ldconfig -p
1094 libs found in the cache '/etc/ld.so.cache'
    libzvbi.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzvbi.so.0
    libzvbi-chains.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzvbi-
chains.so.0
    libzmq.so.5 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzmq.so.5
    libzeitgeist-2.0.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-
gnu/libzeitgeist-2.0.so.0
    (...)
```

Note como a cache emprega o soname completo dos vínculos:

```
$ sudo ldconfig -p |grep libfuse
libfuse.so.2 (libc6,x86-64) => /lib/x86_64-linux-gnu/libfuse.so.2
```

Se fizermos a lista longa de `/lib/x86_64-linux-gnu/libfuse.so.2`, encontraremos a referência ao arquivo-objeto compartilhado real `libfuse.so.2.9.7`, que fica armazenado no mesmo diretório:

```
$ ls -l /lib/x86_64-linux-gnu/libfuse.so.2
lrwxrwxrwx 1 root root 16 Aug 21 2018 /lib/x86_64-linux-gnu/libfuse.so.2 ->
libfuse.so.2.9.7
```

NOTE

Como ele exige acesso de escrita a `/etc/ld.so.cache` (pertencente ao root), você deve estar logado como root ou usar `sudo` para chamar o `ldconfig`. Para saber mais sobre as permissões de `ldconfig`, consulte a página de manual.

Além dos arquivos de configuração descritos acima, a variável de ambiente `LD_LIBRARY_PATH` pode ser usada para adicionar temporariamente novos caminhos para bibliotecas compartilhadas. Ele é composto por um conjunto de diretórios separados por dois pontos (`:`) no qual as bibliotecas são buscadas. Assim, para adicionar `/usr/local/mylib` ao caminho da biblioteca na sessão atual do shell, poderíamos digitar:

```
$ LD_LIBRARY_PATH=/usr/local/mylib
```

Em seguida podemos verificar o valor:

```
$ echo $LD_LIBRARY_PATH
/usr/local/mylib
```

Para adicionar `/usr/local/mylib` ao caminho da biblioteca na sessão atual do shell e exportá-lo para todos os processos secundários originados desse shell, escreveríamos:

```
$ export LD_LIBRARY_PATH=/usr/local/mylib
```

Para remover a variável de ambiente `LD_LIBRARY_PATH`, basta digitar:

```
$ unset LD_LIBRARY_PATH
```

Se quiser tornar as alterações permanentes, escrevemos a linha

```
export LD_LIBRARY_PATH=/usr/local/mylib
```

Em um dos scripts de inicialização do Bash, como `/etc/bash.bashrc` ou `~/.bashrc`.

NOTE

`LD_LIBRARY_PATH` está para as bibliotecas compartilhadas como `PATH` está para os executáveis. Para saber mais sobre variáveis de ambiente e configuração do shell, consulte as lições respectivas.

Buscando pelas dependências de um executável específico

Para buscar as bibliotecas compartilhadas requeridas por um programa específico, use o comando `ldd` seguido do caminho absoluto para o programa. A saída mostra o caminho do arquivo da biblioteca compartilhada, bem como o endereço de memória hexadecimal no qual ele é carregado:

```
$ ldd /usr/bin/git
linux-vdso.so.1 => (0x00007ffcb310000)
libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f18241eb000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f1823fd1000)
libresolv.so.2 => /lib/x86_64-linux-gnu/libresolv.so.2 (0x00007f1823db6000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f1823b99000)
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f1823991000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f18235c7000)
/lib64/ld-linux-x86-64.so.2 (0x00007f182445b000)
```

Da mesma forma, usamos `ldd` para procurar as dependências de um objeto compartilhado:

```
$ ldd /lib/x86_64-linux-gnu/libc.so.6
    /lib64/ld-linux-x86-64.so.2 (0x00007fbfed578000)
    linux-vdso.so.1 (0x00007ffffb7bf5000)
```

Com a opção `-u` (ou `--unused`), o `ldd` imprime as dependências diretas não utilizadas (se existirem):

```
$ ldd -u /usr/bin/git
Unused direct dependencies:
    /lib/x86_64-linux-gnu/libz.so.1
    /lib/x86_64-linux-gnu/libpthread.so.0
    /lib/x86_64-linux-gnu/librt.so.1
```

A razão para haver dependências não utilizadas está relacionada às opções usadas pelo vinculador ao criar o binário. Embora o programa não precise de uma biblioteca não utilizada, ela ainda estava vinculada e rotulada como NEEDED (necessária) nas informações sobre o arquivo-objeto. Para investigar isso, podemos usar comandos como `readelf` ou `objdump`, que serão necessários mais adiante, nos exercícios exploratórios.

Exercícios Guiados

1. Divida os seguintes nomes de bibliotecas compartilhadas nas partes que os constituem:

Nome completo do arquivo	nome da biblioteca	sufixo so	Número da versão
linux-vdso.so.1			
libprocps.so.6			
libdl.so.2			
libc.so.6			
libsystemd.so.0			
ld-linux-x86-64.so.2			

2. Você desenvolveu um programa e deseja adicionar um novo diretório de biblioteca compartilhada em seu sistema (/opt/lib/mylib). Você escreve o caminho absoluto em um arquivo chamado mylib.conf.

- Em que diretório deve ser armazenado esse arquivo?

- Qual comando deve ser executado para que as alterações sejam totalmente efetivas?

3. Qual comando você usaria para listar as bibliotecas compartilhadas exigidas por kill?

Exercícios Exploratórios

1. objdump é um utilitário de linha de comando que exibe informações sobre arquivos objeto. Confira se ele está instalado em seu sistema com `which objdump`. Se não estiver, instale-o agora.

- Use `objdump` com `-p` (ou `--private-headers`) e `grep` para exibir as dependências de `glibc`:

- Use `objdump` com `-p` (ou `--private-headers`) e `grep` para exibir o soname de `glibc`:

- Use `objdump` com `-p` (ou `--private-headers`) e `grep` para exibir as dependências do Bash:

Resumo

Nesta lição, você aprendeu:

- O que é uma biblioteca compartilhada (ou dinâmica).
- As diferenças entre bibliotecas compartilhadas e estáticas.
- Os nomes das bibliotecas compartilhadas (*sonames*).
- Os locais de preferência para as bibliotecas compartilhadas em um sistema Linux, como `/lib` ou `/usr/lib`.
- A finalidade do vinculador dinâmico `ld.so` (ou `ld-linux.so`).
- Como configurar caminhos para as bibliotecas compartilhadas usando arquivos em `/etc/` como `ld.so.conf` ou os arquivos do diretório `ld.so.conf.d`.
- Como configurar caminhos para as bibliotecas compartilhadas usando a variável de ambiente `LD_LIBRARY_PATH`.
- Como buscar por executáveis e dependências de bibliotecas compartilhadas.

Comandos usados nesta lição:

`ls`

Lista o conteúdo do diretório.

`cat`

Concatena arquivos e exibe na saída padrão.

`sudo`

Permite que o superusuário execute o comando com privilégios administrativos.

`ldconfig`

Configura as ligações de tempo de execução do vinculador dinâmico.

`echo`

Exibe o valor da variável de ambiente.

`export`

Exporta o valor da variável de ambiente para shells secundários.

unset

Remove a variável de ambiente.

ldd

Exibe as dependências de objetos compartilhados de um programa.

readelf

Exibe informações sobre arquivos ELF (ELF significa *formato executável e vinculável*).

objdump

Exibe informações de arquivos de objetos.

Respostas aos Exercícios Guiados

1. Divida os seguintes nomes de bibliotecas compartilhadas nas partes que os constituem:

Nome completo do arquivo	nome da biblioteca	sufixo so	Número da versão
linux-vdso.so.1	linux-vdso	.so	1
libprocps.so.6	libprocps	.so	6
libdl.so.2	libdl	.so	2
libc.so.6	libc	.so	6
libsystemd.so.0	libsystemd	.so	0
ld-linux-x86-64.so.2	ld-linux-x86-64	.so	2

2. Você desenvolveu um programa e deseja adicionar um novo diretório de biblioteca compartilhada em seu sistema (/opt/lib/mylib). Você escreve o caminho absoluto em um arquivo chamado mylib.conf.

- Em que diretório deve ser armazenado esse arquivo?

/etc/ld.so.conf.d

- Qual comando deve ser executado para que as alterações sejam totalmente efetivas?

ldconfig

3. Qual comando você usaria para listar as bibliotecas compartilhadas exigidas por kill?

ldd /bin/kill

Respostas aos Exercícios Exploratórios

1. objdump é um utilitário de linha de comando que exibe informações sobre arquivos objeto. Confira se ele está instalado em seu sistema com `which objdump`. Se não estiver, instale-o agora.

- Use `objdump` com `-p` (ou `--private-headers`) e `grep` para exibir as dependências de `glibc`:

```
objdump -p /lib/x86_64-linux-gnu/libc.so.6 | grep NEEDED
```

- Use `objdump` com `-p` (ou `--private-headers`) e `grep` para exibir o soname de `glibc`:

```
objdump -p /lib/x86_64-linux-gnu/libc.so.6 | grep SONAME
```

- Use `objdump` com `-p` (ou `--private-headers`) e `grep` para exibir as dependências do Bash:

```
objdump -p /bin/bash | grep NEEDED
```



102.4 Utilização do sistema de pacotes Debian

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 102.4

Peso

3

Áreas chave de conhecimento

- Instalar, atualizar e desinstalar os pacotes binários Debian.
- Encontrar pacotes contendo um arquivo específico ou bibliotecas que podem estar instaladas ou não.
- Obter informações sobre pacotes como versão, conteúdo, dependências, integridade do pacote e status da instalação (estando o pacote instalado ou não).
- Noções do apt

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- /etc/apt/sources.list
- dpkg
- dpkg-reconfigure
- apt-get
- apt-cache



**Linux
Professional
Institute**

102.4 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	102 Instalação do Linux e gerenciamento de pacotes
Objetivo:	102.4 Como usar o gerenciamento de pacotes do Debian
Lição:	1 de 1

Introdução

Há muito tempo, quando o Linux ainda estava em sua primeira infância, a maneira mais comum de distribuir software era um arquivo comprimido (geralmente um arquivo `.tar.gz`) com código-fonte que o usuário precisava descomprimir e compilar.

No entanto, à medida que aumentava a quantidade e a complexidade do software, ficou clara a necessidade de uma maneira de distribuir software pré-compilado. Afinal, nem todos dispunham dos recursos necessários, tanto de tempo quanto de poder de computação, para compilar projetos grandes como o kernel do Linux ou um X Server.

Nasceu assim uma busca para padronizar uma maneira de distribuir esses “pacotes” de software, e os primeiros gerenciadores de pacotes nasceram. Essas ferramentas facilitaram muito a tarefa de instalar, configurar ou remover software de um sistema.

Um deles era o formato de pacote Debian (`.deb`) e sua ferramenta de pacote (`dpkg`). Hoje, eles são amplamente utilizados não somente no próprio Debian, mas também em seus derivativos, como o Ubuntu e os derivados dele.

Outra ferramenta de gerenciamento de pacotes muito popular nos sistemas baseados em Debian é a *Advanced Package Tool* (`apt`), capaz de otimizar muitos dos aspectos da instalação, manutenção e remoção de pacotes.

Nesta lição, aprenderemos como usar o `dpkg` e o `apt` para obter, instalar, manter e remover software de um sistema Linux baseado em Debian.

A ferramenta de pacotes do Debian (`dpkg`)

A ferramenta *Debian Package* (`dpkg`) é o utilitário essencial para instalar, configurar, manter e remover pacotes de software em sistemas baseados em Debian. A operação mais básica é instalar um pacote `.deb`, o que pode ser feito com:

```
# dpkg -i PACKAGENAME
```

Onde `PACKAGENAME` é o nome do arquivo `.deb` que desejamos instalar.

As atualizações de pacotes são tratadas da mesma maneira. Antes de instalar um pacote, o `dpkg` verifica se já existe uma versão anterior no sistema. Nesse caso, o pacote será atualizado para a nova versão. Caso contrário, uma nova cópia será instalada.

Como gerenciar as dependências

Na maioria das vezes, um pacote depende de outros para funcionar adequadamente. Por exemplo, um editor de imagens pode precisar de bibliotecas para abrir arquivos JPEG ou outro utilitário pode precisar de um kit de ferramentas de widget como o Qt ou o GTK para a interface de usuário.

O `dpkg` verifica se essas dependências estão instaladas em seu sistema; se não estiverem, ele não conseguirá instalar o pacote. Nesse caso, o `dpkg` lista os pacotes faltantes. No entanto, ele *não pode* resolver dependências sozinho. Cabe ao usuário encontrar os pacotes `.deb` com as dependências correspondentes e instalá-los.

No exemplo abaixo, o usuário tentou instalar o pacote do editor de vídeo OpenShot, mas algumas dependências estavam ausentes:

```
# dpkg -i openshot-qt_2.4.3+dfsg1-1_all.deb
(Reading database ... 269630 files and directories currently installed.)
Preparing to unpack openshot-qt_2.4.3+dfsg1-1_all.deb ...
Unpacking openshot-qt (2.4.3+dfsg1-1) over (2.4.3+dfsg1-1) ...
dpkg: dependency problems prevent configuration of openshot-qt:
  openshot-qt depends on fonts-cantarell; however:
```

```

Package fonts-cantarell is not installed.
openshot-qt depends on python3-openshot; however:
  Package python3-openshot is not installed.
openshot-qt depends on python3-pyqt5; however:
  Package python3-pyqt5 is not installed.
openshot-qt depends on python3-pyqt5.qtsvg; however:
  Package python3-pyqt5.qtsvg is not installed.
openshot-qt depends on python3-pyqt5.qtwebkit; however:
  Package python3-pyqt5.qtwebkit is not installed.
openshot-qt depends on python3-zmq; however:
  Package python3-zmq is not installed.

dpkg: error processing package openshot-qt (--install):
  dependency problems - leaving unconfigured
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for gnome-menus (3.32.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-4ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for man-db (2.8.5-2) ...
Errors were encountered while processing:
  openshot-qt

```

Como mostrado acima, o OpenShot depende dos pacotes fonts-cantarell, python3-openshot, python3-pyqt5, python3-pyqt5.qtsvg, python3-pyqt5.qtwebkit e python3-zmq. Todos eles precisam ser instalados antes que a instalação do OpenShot possa ser realizada com sucesso.

Removendo pacotes

Para remover um pacote, passe o parâmetro `-r` para `dpkg`, seguido pelo nome do pacote. Por exemplo, o seguinte comando removerá o pacote `unrar` do sistema:

```

# dpkg -r unrar
(Reading database ... 269630 files and directories currently installed.)
Removing unrar (1:5.6.6-2) ...
Processing triggers for man-db (2.8.5-2) ...

```

A operação de remoção também executa uma verificação de dependências. Um pacote não pode ser removido, a menos que todos os outros pacotes que dependem dele também o sejam. Se você tentar fazer isso, receberá uma mensagem de erro como a abaixo:

```

# dpkg -r p7zip
dpkg: dependency problems prevent removal of p7zip:

```

```
winetricks depends on p7zip; however:
Package p7zip is to be removed.
p7zip-full depends on p7zip (= 16.02+dfsg-6).
```

```
dpkg: error processing package p7zip (--remove):
dependency problems – not removing
Errors were encountered while processing:
p7zip
```

É possível passar vários nomes de pacotes para o `dpkg -r`, para que todos sejam removidos de uma só vez.

Quando um pacote é removido, os arquivos de configuração correspondentes são deixados no sistema. Se a ideia for remover tudo que esteja associado ao pacote, use a opção `-P` (purge) em vez de `-r`.

NOTE

Podemos forçar o `dpkg` a instalar ou remover um pacote, mesmo que as dependências não sejam atendidas, adicionando o parâmetro `--force` como em `dpkg -i --force PACKAGENAME`. No entanto, isso provavelmente deixará o pacote, ou mesmo o sistema, em um estado de falha. *Não use `--force`, a menos que tenha certeza absoluta do que está fazendo.*

Obtendo informações sobre um pacote

Para obter informações sobre um pacote `.deb`, como a versão, arquitetura, mantenedor, dependências e outras, use o comando `dpkg` com o parâmetro `-I`, seguido pelo nome do arquivo do pacote que deseja inspecionar:

```
# dpkg -I google-chrome-stable_current_amd64.deb
new Debian package, version 2.0.
size 59477810 bytes: control archive=10394 bytes.
    1222 bytes,   13 lines      control
   16906 bytes,   457 lines    * postinst          #!/bin/sh
   12983 bytes,   344 lines    * postrm          #!/bin/sh
     1385 bytes,   42 lines    * prerm          #!/bin/sh
Package: google-chrome-stable
Version: 76.0.3809.100-1
Architecture: amd64
Maintainer: Chrome Linux Team <chromium-dev@chromium.org>
Installed-Size: 205436
Pre-Depends: dpkg (>= 1.14.0)
Depends: ca-certificates, fonts-liberation, libappindicator3-1, libasound2 (>=
```

```
1.0.16), libatk-bridge2.0-0 (>= 2.5.3), libatk1.0-0 (>= 2.2.0), libatspi2.0-0 (>=
2.9.90), libc6 (>= 2.16), libcairo2 (>= 1.6.0), libcups2 (>= 1.4.0), libdbus-1-3
(>= 1.5.12), libexpat1 (>= 2.0.1), libgcc1 (>= 1:3.0), libgdk-pixbuf2.0-0 (>=
2.22.0), libglib2.0-0 (>= 2.31.8), libgtk-3-0 (>= 3.9.10), libnspr4 (>= 2:4.9-2~),
libnss3 (>= 2:3.22), libpango-1.0-0 (>= 1.14.0), libpangocairo-1.0-0 (>= 1.14.0),
libuuid1 (>= 2.16), libx11-6 (>= 2:1.4.99.1), libx11-xcb1, libxcb1 (>= 1.6),
libxcomposite1 (>= 1:0.3-1), libxcursor1 (>> 1.1.2), libxdamage1 (>= 1:1.1),
libxext6, libxfixes3, libxi6 (>= 2:1.2.99.4), libxrandr2 (>= 2:1.2.99.3),
libxrender1, libxss1, libxtst6, lsb-release, wget, xdg-utils (>= 1.0.2)
```

Recommends: libu2f-udev

Provides: www-browser

Section: web

Priority: optional

Description: The web browser from Google

Google Chrome is a browser that combines a minimal design with sophisticated technology to make the web faster, safer, and easier.

Listando os pacotes instalados e conteúdos dos pacotes

Para obter uma lista de todos os pacotes instalados no sistema, use a opção `--get-selections`, como em `dpkg --get-selections`. Também é possível obter uma lista de todos os arquivos instalados por um pacote específico passando o parâmetro `-L PACKAGE_NAME` para o `dpkg`, como mostrado abaixo:

```
# dpkg -L unrar
/.
/usr
/usr/bin
/usr/bin/unrar-nonfree
/usr/share
/usr/share/doc
/usr/share/doc/unrar
/usr/share/doc/unrar/changelog.Debian.gz
/usr/share/doc/unrar/copyright
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/unrar-nonfree.1.gz
```

Descobrindo qual pacote possui um arquivo específico

Às vezes, podemos precisar descobrir qual pacote possui um arquivo específico no sistema. Para isso, usamos o utilitário `dpkg-query`, seguido pelo parâmetro `-S` e o caminho para o arquivo em questão:

```
# dpkg-query -S /usr/bin/unrar-nonfree
unrar: /usr/bin/unrar-nonfree
```

Reconfigurando os pacotes instalados

Quando um pacote é instalado, existe uma etapa de configuração chamada *pós-instalação*, na qual um script é executado para definir tudo o que é necessário para a execução do software, como permissões, local dos arquivos de configuração etc. Ele também pode fazer algumas perguntas ao usuário sobre as preferências para a execução do software.

Às vezes, devido a um arquivo de configuração corrompido ou malformado, pode ser necessário restaurar as configurações de um pacote para o estado “novo”, ou você pode querer alterar as respostas que deu às questões da configuração inicial. Para isso, execute o utilitário `dpkg-reconfigure`, seguido pelo nome do pacote.

Este programa faz um backup dos arquivos de configuração antigos, descompacta os novos nos diretórios corretos e executa o script de *pós-instalação* fornecido pelo pacote, como se o pacote tivesse sido instalado pela primeira vez. Tente reconfigurar o pacote `tzdata` com o seguinte exemplo:

```
# dpkg-reconfigure tzdata
```

Ferramenta de pacotes avançada (apt)

O *Advanced Package Tool* (APT) é um sistema de gerenciamento de pacotes com um conjunto de ferramentas que simplifica bastante a instalação, atualização, remoção e gerenciamento de pacotes. O APT fornece recursos como busca avançada e resolução automática de dependências.

O APT não é um “substituto” para o `dpkg`. Ele está mais para um “front end”, simplificando operações e preenchendo lacunas na funcionalidade do `dpkg`, como a resolução de dependências.

O APT trabalha em conjunto com repositórios de software que contêm os pacotes disponíveis para instalação. Esses repositórios podem ser um servidor local ou remoto, ou (o que é menos comum) até um disco CD-ROM.

As distribuições Linux, como Debian e Ubuntu, mantêm seus próprios repositórios. Outros repositórios podem ser mantidos por desenvolvedores ou grupos de usuários para fornecer software que não está disponível nos principais repositórios de distribuição. Existem muitos utilitários que interagem com o APT, sendo os principais:

apt-get

usado para baixar, instalar, atualizar ou remover pacotes do sistema.

apt-cache

usado para executar operações, como pesquisas, no índice do pacote.

apt-file

usado para buscar arquivos dentro de pacotes.

Existe também um utilitário mais “amigável” chamado simplesmente `apt`, que combina as opções mais usadas de `apt-get` e `apt-cache` em um único utilitário. Muitos dos comandos de `apt` são os mesmos de `apt-get`, de forma que eles são intercambiáveis em diversos casos. Porém, como o `apt` pode não estar instalado em um sistema, é recomendável aprender a usar o `apt-get` e o `apt-cache`.

NOTE

É melhor usar `apt` e `apt-get` com uma conexão de rede, pois pode ser necessário baixar pacotes e índices de pacotes de um servidor remoto.

Atualizando o índice do pacote

Antes de instalar ou atualizar o software com o APT, é recomendável atualizar primeiro o índice do pacote para recuperar informações sobre pacotes novos e atualizados. Isso é feito com o comando `apt-get`, seguido pelo parâmetro `update`:

```
# apt-get update
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 https://repo.skype.com/deb stable InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu disco InRelease
Hit:4 http://repository.spotify.com stable InRelease
Hit:5 http://dl.google.com/linux/chrome/deb stable Release
Hit:6 http://apt.pop-os.org/proprietary disco InRelease
Hit:7 http://ppa.launchpad.net/system76/pop/ubuntu disco InRelease
Hit:8 http://us.archive.ubuntu.com/ubuntu disco-security InRelease
Hit:9 http://us.archive.ubuntu.com/ubuntu disco-updates InRelease
Hit:10 http://us.archive.ubuntu.com/ubuntu disco-backports InRelease
Reading package lists... Done
```

TIP

Em vez de `apt-get update`, você também pode usar `apt update`.

Instalando e removendo pacotes

Com o índice do pacote atualizado, já podemos instalar o pacote. Isso é feito com `apt-get install`,

seguido pelo nome do pacote a instalar:

```
# apt-get install xournal
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  xournal
0 upgraded, 1 newly installed, 0 to remove and 75 not upgraded.
Need to get 285 kB of archives.
After this operation, 1041 kB of additional disk space will be used.
```

Da mesma maneira use apt-get remove, seguido pelo nome do pacote:

```
# apt-get remove xournal
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  xournal
0 upgraded, 0 newly installed, 1 to remove and 75 not upgraded.
After this operation, 1041 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Lembre-se de que, ao instalar ou remover pacotes, o APT executará a resolução automática de dependências. Assim, todos os pacotes adicionais necessários para o pacote que você está instalando *também serão instalados*, e os pacotes que dependem do pacote que você está removendo *também serão removidos*. O APT sempre mostra o que será instalado ou removido antes de perguntar se você deseja continuar:

```
# apt-get remove p7zip
Reading package lists... Done
Building dependency tree
The following packages will be REMOVED:
  android-libbacktrace android-libunwind android-libutils
  android-libziparchive android-sdk-platform-tools fastboot p7zip p7zip-full
0 upgraded, 0 newly installed, 8 to remove and 75 not upgraded.
After this operation, 6545 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Note que, quando um pacote é removido, os arquivos de configuração correspondentes são deixados

no sistema. Para remover o pacote e todos os arquivos de configuração, use o parâmetro `purge` em vez de `remove`, ou o parâmetro `remove` com a opção `--purge`:

```
# apt-get purge p7zip
```

ou

```
# apt-get remove --purge p7zip
```

TIP Também podemos usar `apt install` e `apt remove`.

Corrigindo dependências quebradas

É possível ter “dependências quebradas” em um sistema. O termo significa que um ou mais dos pacotes instalados dependem de outros pacotes que não foram instalados ou não estão mais presentes. Isso pode ocorrer devido a um erro do APT ou a um pacote instalado manualmente.

Para resolver o problema, use o comando `apt-get install -f`. Ele procura “consertar” os pacotes quebrados instalando as dependências ausentes, garantindo que todos os pacotes voltem a ficar consistentes.

TIP Você também pode usar `apt install -f`.

Atualizando pacotes

O APT pode ser usado para atualizar automaticamente quaisquer pacotes instalados para as versões mais recentes disponíveis nos repositórios. Isso é feito com o comando `apt-get upgrade`. Antes de executá-lo, primeiro atualize o índice do pacote com `apt-get update`:

```
# apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu disco InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu disco-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu disco-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu disco-backports InRelease
Reading package lists... Done

# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```

Calculating upgrade... Done
The following packages have been kept back:
  gnome-control-center
The following packages will be upgraded:
  cups cups-bsd cups-client cups-common cups-core-drivers cups-daemon
  cups-ipp-utils cups-ppdc cups-server-common firefox-locale-ar (...)

74 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Need to get 243 MB of archives.
After this operation, 30.7 kB of additional disk space will be used.
Do you want to continue? [Y/n]

```

O resumo na parte inferior da saída mostra quantos pacotes serão atualizados, quantos serão instalados, removidos ou retidos, o tamanho total do download e quanto espaço extra em disco será necessário para concluir a operação. Para concluir a atualização, basta responder `Y` e aguardar o `apt-get` concluir a tarefa.

Para atualizar um único pacote, basta executar `apt-get upgrade` seguido pelo nome do pacote. Como o `dpkg`, o `apt-get` primeiro verifica se uma versão anterior de um pacote está instalada. Nesse caso, o pacote será atualizado para a versão mais recente disponível no repositório. Caso contrário, uma cópia nova será instalada.

TIP | Você também pode usar `apt upgrade` e `apt update`.

A cache local

Quando instalamos ou atualizamos um pacote, o arquivo `.deb` correspondente é baixado em um diretório de cache local antes do pacote ser instalado. Por padrão, esse diretório é `/var/cache/apt/archives`. Os arquivos parcialmente baixados são copiados para `/var/cache/apt/archives/partial/`.

Conforme instalamos e atualizamos diferentes pacotes, o diretório da cache pode ficar bastante volumoso. Para recuperar espaço, podemos esvaziar a cache usando o comando `apt-get clean`. Ele remove o conteúdo dos diretórios `/var/cache/apt/archives` e `/var/cache/apt/archives/partial/`.

TIP | Também podemos usar `apt clean`.

Buscando pacotes

O utilitário `apt-cache` pode ser usado para executar operações no índice dos pacotes, como buscar um pacote específico ou listar quais pacotes contêm um arquivo determinado.

Para realizar uma pesquisa, use `apt-cache search` seguido por um padrão de pesquisa. A saída será uma lista de todos os pacotes que contêm o padrão, seja no nome do pacote, na descrição ou nos arquivos fornecidos.

```
# apt-cache search p7zip
liblzma-dev - XZ-format compression library - development files
liblzma5 - XZ-format compression library
forensics-extra - Forensics Environment - extra console components (metapackage)
p7zip - 7zr file archiver with high compression ratio
p7zip-full - 7z and 7za file archivers with high compression ratio
p7zip-rar - non-free rar module for p7zip
```

No exemplo acima, a entrada `liblzma5 - XZ-format compression library` não parece corresponder ao padrão. No entanto, se mostrarmos as informações completas do pacote, incluindo a descrição, usando o parâmetro `show`, encontraremos o padrão:

```
# apt-cache show liblzma5
Package: liblzma5
Architecture: amd64
Version: 5.2.4-1
Multi-Arch: same
Priority: required
Section: libs
Source: xz-utils
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Jonathan Nieder <jrnieder@gmail.com>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 259
Depends: libc6 (>= 2.17)
Breaks: liblzma2 (<< 5.1.1alpha+20110809-3~)
Filename: pool/main/x/xz-utils/liblzma5_5.2.4-1_amd64.deb
Size: 92352
MD5sum: 223533a347dc76a8cc9445fcfc6146ec3
SHA1: 8ed14092fb1caecfebc556fda0745e1e74ba5a67
SHA256: 01020b5a0515dbc9a7c00b464a65450f788b0258c3fbb733ecad0438f5124800
Homepage: https://tukaani.org/xz/
Description-en: XZ-format compression library
XZ is the successor to the Lempel-Ziv/Markov-chain Algorithm
compression format, which provides memory-hungry but powerful
compression (often better than bzip2) and fast, easy decompression.
.
The native format of liblzma is XZ; it also supports raw (headerless)
```

streams and the older LZMA format used by lzma. (For 7-Zip's related format, use the `p7zip` package instead.)

Você pode usar *expressões regulares* com o padrão de pesquisa, permitindo buscas bastante complexas (e precisas). No entanto, esse tópico está fora do escopo desta lição.

TIP Também é possível usar `apt search` em vez de `apt-cache search` e `apt show` em vez de `apt-cache show`.

A lista de fontes

O APT usa uma lista de fontes para saber de onde obter pacotes. Esta lista é armazenada no arquivo `sources.list`, localizada dentro do diretório `/etc/apt`. Esse arquivo pode ser editado diretamente com um editor de texto, como `vi`, `pico` ou `nano`, ou com utilitários gráficos como `aptitude` ou `synaptic`.

Uma linha típica dentro de `sources.list` é assim:

```
deb http://us.archive.ubuntu.com/ubuntu/ disco main restricted universe multiverse
```

A sintaxe é tipo de arquivo, URL, distribuição e um ou mais componentes, onde:

Tipo de arquivo

Um repositório pode conter pacotes com software pronto para execução (pacotes binários, do tipo `deb`) ou com o código fonte desse software (pacotes de origem, do tipo `deb-src`). O exemplo acima fornece pacotes binários.

URL

A URL do repositório.

Distribuição

O nome (ou codinome) da distribuição para a qual os pacotes são fornecidos. Um repositório pode hospedar pacotes para várias distribuições. No exemplo acima, `disco` é o codinome do Ubuntu 19.04, *Disco Dingo*.

Componentes

Cada componente representa um conjunto de pacotes. Esses componentes podem ser diferentes nas diversas distribuições Linux. Por exemplo, no Ubuntu e derivados, eles são:

main

contém pacotes de código aberto oficialmente suportados.

restricted

contém software de código fechado oficialmente suportado, como drivers de dispositivo para placas gráficas, por exemplo.

universe

contém software de código aberto mantido pela comunidade.

multiverse

contém software não suportado, de código fechado ou protegido por patente.

No Debian, os principais componentes são:

main

consiste em pacotes compatíveis com as *Debian Free Software Guidelines* (DFSG), que não dependem de software externos a essa área para operar. Os pacotes incluídos aqui são considerados parte da distribuição Debian.

contrib

contém pacotes compatíveis com DFSG, mas que dependem de outros pacotes que não estão no main.

non-free

contém pacotes que não são compatíveis com o DFSG.

security

contém atualizações de segurança.

backports

contém versões mais recentes dos pacotes que estão em main. O ciclo de desenvolvimento das versões estáveis do Debian é bastante longo (cerca de dois anos), garantindo que os usuários possam obter os pacotes mais atualizados sem precisar modificar o repositório principal main.

NOTE

Para saber mais sobre as *Debian Free Software Guidelines*, visite:
https://www.debian.org/social_contract#guidelines

Para adicionar um novo repositório do qual obter pacotes, você pode simplesmente acrescentar a linha correspondente (geralmente fornecida pelo mantenedor do repositório) ao final de sources.list, salvar o arquivo e recarregar o índice do pacote com apt-get update. Depois disso, os

pacotes do novo repositório estarão disponíveis para instalação usando `apt-get install`.

Lembre-se de que as linhas que começam com o caractere `#` são consideradas comentários e ignoradas. ===== O diretório `/etc/apt/sources.list.d`

Dentro do diretório `/etc/apt/sources.list.d`, podemos adicionar arquivos com repositórios adicionais a serem usados pelo APT, sem a necessidade de modificar o arquivo principal `/etc/apt/sources.list`. Trata-se de arquivos de texto simples, com a mesma sintaxe descrita acima e a extensão de arquivo `.list`.

Abaixo, vemos o conteúdo de um arquivo chamado `/etc/apt/sources.list.d/buster-backports.list`:

```
deb http://deb.debian.org/debian buster-backports main contrib non-free
deb-src http://deb.debian.org/debian buster-backports main contrib non-free
```

Listando o conteúdo dos pacotes e buscando arquivos

Um utilitário chamado `apt-file` pode ser usado para executar mais operações no índice do pacote, como listar o conteúdo de um pacote ou localizar um pacote que contenha um arquivo específico. Esse utilitário talvez não esteja instalado por padrão no seu sistema. Nesse caso, geralmente é possível instalá-lo usando `apt-get`:

```
# apt-get install apt-file
```

Após a instalação, será preciso atualizar o cache do pacote usado para `apt-file`:

```
# apt-file update
```

Isso geralmente leva apenas alguns segundos. Depois disso, você estará pronto para usar o `apt-file`.

Para listar o conteúdo de um pacote, use o parâmetro `list` seguido pelo nome do pacote:

```
# apt-file list unrar
unrar: /usr/bin/unrar-nonfree
unrar: /usr/share/doc/unrar/changelog.Debian.gz
unrar: /usr/share/doc/unrar/copyright
unrar: /usr/share/man/man1/unrar-nonfree.1.gz
```

TIP Também podemos usar `apt list` em vez de `apt-file list`.

Você pode procurar um arquivo em todos os pacotes usando o parâmetro `search`, seguido pelo nome do arquivo. Por exemplo, se quisermos saber qual pacote fornece um arquivo chamado `libSDL2.so`, usaremos:

```
# apt-file search libSDL2.so
libsdl2-dev: /usr/lib/x86_64-linux-gnu/libSDL2.so
```

A resposta é o pacote `libsdl2-dev`, que fornece o arquivo `/usr/lib/x86_64-linux-gnu/libSDL2.so`.

A diferença entre `apt-file search` e `dpkg-query` é que `apt-file search` também busca por pacotes não instalados, ao passo que `dpkg-query` mostra somente os arquivos pertencentes a um pacote instalado.

Exercícios Guiados

1. Qual seria o comando para instalar um pacote chamado `package.deb` usando `dpkg`?

2. Usando `dpkg-query`, descubra qual pacote contém um arquivo chamado `7zr.1.gz`.

3. É possível remover um pacote chamado `unzip` do sistema usando `dpkg -r unzip` se o pacote `file-roller` depender dele? Se não, qual seria o jeito correto de fazer isso?

4. Usando o utilitário `apt-file`, como podemos descobrir qual pacote contém o arquivo `/usr/bin/unrar`?

5. Usando o `apt-cache`, qual seria o comando para exibir informações para o pacote `gimp`?

Exercícios Exploratórios

1. Considere um repositório com pacotes de fontes Debian para a distribuição `xenial`, hospedado em `http://us.archive.ubuntu.com/ubuntu/` e com pacotes para o componente `universe`. Qual seria a linha correspondente a adicionar a `/etc/apt/sources.list`?

2. Ao compilar um programa, aparece uma mensagem de erro reclamando que o arquivo de cabeçalho `zzip-io.h` não está presente no seu sistema. Como você pode descobrir qual pacote fornece esse arquivo?

3. Como podemos ignorar um aviso de dependência e remover um pacote usando `dpkg`, mesmo que haja pacotes que dependam dele no sistema?

4. Como é possível obter mais informações sobre um pacote chamado `midori` usando `apt`?

5. Antes de instalar ou atualizar pacotes com o `apt`, qual comando deve ser usado para garantir que o índice do pacote esteja atualizado?

Resumo

Nesta lição, você aprendeu:

- Como usar o `dpkg` para instalar e remover pacotes.
- Como listar os pacotes instalados e o conteúdo do pacote.
- Como reconfigurar um pacote instalado.
- O que é o `apt` e como usá-lo para instalar, atualizar e remover pacotes.
- Como usar o `apt-cache` para procurar pacotes.
- Como funciona o arquivo `/etc/apt/sources.list`.
- Como usar o `apt-file` para mostrar o conteúdo de um pacote, ou como encontrar qual pacote contém um arquivo específico.

Os seguintes comandos foram abordados:

`dpkg -i`

Instala um único pacote ou uma lista de pacotes separados por espaço.

`dpkg -r`

Remove um pacote ou uma lista de pacotes separados por espaço.

`dpkg -I`

Inspeciona um pacote, fornecendo detalhes sobre o software instalado e todas as dependências necessárias.

`dpkg --get-selections`

Lista todos os pacotes que o `dpkg` instalou no sistema.

`dpkg -L`

Imprime uma lista de todos os arquivos instalados por um pacote específico.

`dpkg-query`

Com um nome de arquivo especificado, este comando imprimirá o pacote que instalou o arquivo.

`dpkg-reconfigure`

Este comando re-executa um script *pós-instalação* de pacotes para que um administrador possa fazer ajustes de configuração da instalação do pacote.

apt-get update

Este comando atualiza o índice do pacote local de acordo com o que está disponível nos repositórios configurados no diretório /etc/apt/.

apt-get install

Este comando baixa um pacote de um repositório remoto e o instala junto com suas dependências. Também pode ser usado para instalar um pacote Debian que já foi baixado.

apt-get remove

Este comando desinstala o(s) pacote(s) especificado(s) do sistema.

apt-cache show

Assim como o comando dpkg -I, pode ser usado para exibir os detalhes de um pacote específico.

apt-cache search

Este comando procura um pacote específico no banco de dados em cache local do APT.

apt-file update

Este comando atualiza o cache do pacote para que o apt-file possa consultar seu conteúdo.

apt-file search

Este comando procura em qual pacote um arquivo está incluído. Uma lista de todos os pacotes que contêm o padrão é retornada.

apt-file list

Este comando é usado para listar o conteúdo de um pacote, assim como o comando dpkg -L.

Respostas aos Exercícios Guiados

- Qual seria o comando para instalar um pacote chamado `package.deb` usando `dpkg`?

Passe o parâmetro `-i` para o `dpkg`:

```
# dpkg -i package.deb
```

- Usando `dpkg-query`, descubra qual pacote contém um arquivo chamado `7zr.1.gz`.

Adicione o parâmetro `-S` a `dpkg-query`:

```
# dpkg-query -S 7zr.1.gz
```

- É possível remover um pacote chamado `unzip` do sistema usando `dpkg -r unzip` se o pacote `file-roller` depender dele? Se não, qual seria o jeito correto de fazer isso?

Não. O `dpkg` não resolve dependências e não permite remover um pacote se outro pacote instalado depender dele. Neste exemplo, podemos primeiro remover `file-roller` (pressupondo que nada depende dele) e em seguida remover `unzip`, ou remover os dois ao mesmo tempo com

```
# dpkg -r unzip file-roller
```

- Usando o utilitário `apt-file`, como podemos descobrir qual pacote contém o arquivo `/usr/bin/unrar`?

Use o parâmetro `search` seguido pelo caminho (ou nome de arquivo):

```
# apt-file search /usr/bin/unrar
```

- Usando o `apt-cache`, qual seria o comando para exibir informações para o pacote `gimp`?

Use o parâmetro `show` seguido pelo nome do pacote:

```
# apt-cache show gimp
```

Respostas aos Exercícios Exploratórios

1. Considere um repositório com pacotes de fontes Debian para a distribuição `xenial`, hospedado em `http://us.archive.ubuntu.com/ubuntu/` e com pacotes para o componente `universe`. Qual seria a linha correspondente a adicionar a `/etc/apt/sources.list`?

Os pacotes fonte são do tipo `deb-src`, então a linha deve ser:

```
deb-src http://us.archive.ubuntu.com/ubuntu/ xenial universe
```

Essa linha também poderia ser adicionada a um arquivo `.list` em `/etc/apt/sources.list.d/`. Ela pode ter qualquer nome, mas é melhor que seja descritivo, algo como `xenial_sources.list`.

2. Ao compilar um programa, aparece uma mensagem de erro reclamando que o arquivo de cabeçalho `zzip-io.h` não está presente no seu sistema. Como você pode descobrir qual pacote fornece esse arquivo?

Use `apt-file search` para descobrir qual pacote contém um arquivo que não está presente no sistema:

```
# apt-file search zzip-io.h
```

3. Como podemos ignorar um aviso de dependência e remover um pacote usando `dpkg`, mesmo que haja pacotes que dependam dele no sistema?

O parâmetro `--force` poderia ser usado, mas isso *jamais* deve ser feito a menos que se saiba exatamente o que se está fazendo, já que existe um risco enorme de que o sistema seja deixado em um estado inconsistente ou “quebrado”.

4. Como é possível obter mais informações sobre um pacote chamado `midori` usando `apt`?

Use `apt-cache show` seguido pelo nome do pacote:

```
# apt-cache show midori
```

5. Antes de instalar ou atualizar pacotes com o `apt`, qual comando deve ser usado para garantir que o índice do pacote esteja atualizado?

`apt-get update` deve ser usado. Ele baixa os índices mais recentes do pacote dos repositórios

descritos no arquivo `/etc/apt/sources.list` ou no diretório `/etc/apt/sources.list.d/`.



102.5 Utilização do sistema de pacotes RPM e YUM

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 102.5

Peso

3

Áreas chave de conhecimento

- Instalar, reinstalar, atualizar e remover pacotes usando RPM, YUM e Zypper.
- Obter informações dos pacotes RPM tais como versão, status, dependências, integridade e assinaturas.
- Determinar quais arquivos um pacote fornece, bem como encontrar de qual pacote um arquivo específico vem.
- Noções do dnf.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- rpm
- rpm2cpio
- /etc/yum.conf
- /etc/yum.repos.d/
- yum
- zypper



102.5 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	102 Instalação do Linux e gerenciamento de pacotes
Objetivo:	102.5 Uso e gerenciamento de pacotes com RPM e YUM
Lição:	1 de 1

Introdução

Há muito tempo, quando o Linux ainda estava em sua primeira infância, a maneira mais comum de distribuir software era um arquivo comprimido (geralmente um arquivo `.tar.gz`) com código-fonte que o usuário precisava descomprimir e compilar.

No entanto, à medida que aumentava a quantidade e a complexidade do software, ficou clara a necessidade de uma maneira de distribuir software pré-compilado. Afinal, nem todos dispunham dos recursos necessários, tanto de tempo quanto de poder de computação, para compilar projetos grandes como o kernel do Linux ou um X Server.

Nasceu assim uma busca para padronizar uma maneira de distribuir esses “pacotes” de software, e os primeiros gerenciadores de pacotes nasceram. Essas ferramentas facilitaram muito a tarefa de instalar, configurar ou remover software de um sistema.

Um deles era o *RPM Package Manager* e sua ferramenta de pacote (`rpm`), desenvolvida pela Red Hat. Hoje, eles são amplamente utilizados não somente no próprio Red Hat Enterprise Linux (RHEL), mas também em seus descendentes, como o Fedora, CentOS e Oracle Linux, outras distribuições como o

openSUSE e até mesmo outros sistemas operacionais, como o AIX da IBM.

Outras ferramentas de gerenciamento de pacotes populares nas distribuições compatíveis com o Red Hat são o `yum` (YellowDog Updater Modified), o `dnf` (Dandified YUM) e o `zypper`, capazes de otimizar muitos dos aspectos da instalação, manutenção e remoção de pacotes, facilitando bastante o gerenciamento deles.

Nesta lição, você aprenderá a usar o `rpm`, o `yum`, o `dnf` e o `zypper` para obter, instalar, gerenciar e remover software em um sistema Linux.

NOTE

Apesar de usar o mesmo formato de pacote, existem diferenças internas entre as distribuições, de modo que um pacote feito especificamente para o openSUSE pode não funcionar em um sistema RHEL e vice-versa. Ao procurar por pacotes, verifique sempre a compatibilidade e, se possível, tente encontrar um que seja personalizado para sua distribuição específica.

O RPM Package Manager (`rpm`)

O RPM Package Manager (`rpm`) é a ferramenta essencial para gerenciar pacotes de software em sistemas baseados (ou derivados) no Red Hat.

Instalando, atualizando e removendo pacotes

A operação mais básica é a instalação de um pacote, o que pode ser feito com:

```
# rpm -i PACKAGENAME
```

Onde `PACKAGENAME` é o nome do pacote `.rpm` que se deseja instalar.

Se existir uma versão anterior de um pacote no sistema, será possível atualizar para uma versão mais recente usando o parâmetro `-U`:

```
# rpm -U PACKAGENAME
```

Se não houver uma versão anterior do `PACKAGENAME`, uma nova cópia será instalada. Para evitar isso e *apenas* atualizar um pacote que esteja *instalado*, use a opção `-F`.

Em ambas as operações, o parâmetro `-v` permite obter uma saída detalhada (mais informações são mostradas durante a instalação) e `-h` permite exibir símbolos de hash (#) como ajuda visual para acompanhar o progresso da instalação. Vários parâmetros podem ser combinados em um; portanto,

`rpm -i -v -h` é o mesmo que `rpm -ivh`. Para remover um pacote instalado, passe o parâmetro `-e` (como em “erase”, apagar) para o `rpm`, seguido pelo nome do pacote que se deseja remover:

```
# rpm -e wget
```

Se um pacote instalado depender do pacote que está sendo removido, aparecerá uma mensagem de erro:

```
# rpm -e unzip
error: Failed dependencies:
/usr/bin/unzip is needed by (installed) file-roller-3.28.1-2.el7.x86_64
```

Para concluir a operação, primeiro é necessário remover os pacotes que dependem daquele que se deseja remover (no exemplo acima, `file-roller`). Podemos passar vários nomes de pacotes para o `rpm -e` para remover vários pacotes de uma só vez.

Como gerenciar as dependências

Muitas vezes, um pacote pode depender de outros para funcionar como pretendido. Por exemplo, um editor de imagens pode precisar de bibliotecas para abrir arquivos JPG, ou um utilitário pode precisar de um kit de ferramentas de widget como o Qt ou o GTK para sua interface de usuário.

O `rpm` verifica se essas dependências estão instaladas em seu sistema; se não estiverem, ele não conseguirá instalar o pacote. Nesse caso, o `rpm` lista os pacotes faltantes. No entanto, ele *não pode* resolver dependências sozinho.

No exemplo abaixo, o usuário tentou instalar o pacote do editor de imagens GIMP, mas algumas dependências estavam ausentes:

```
# rpm -i gimp-2.8.22-1.el7.x86_64.rpm
error: Failed dependencies:
babl(x86-64) >= 0.1.10 is needed by gimp-2:2.8.22-1.el7.x86_64
gegl(x86-64) >= 0.2.0 is needed by gimp-2:2.8.22-1.el7.x86_64
gimp-libs(x86-64) = 2:2.8.22-1.el7 is needed by gimp-2:2.8.22-1.el7.x86_64
libbabl-0.1.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgegl-0.2.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimp-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpbase-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpcolor-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpconfig-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
```

```
libgimpmath-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpmodule-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpmath-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpui-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libgimpwidgets-2.0.so.0()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libmng.so.1()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libwmf-0.2.so.7()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
libwmflite-0.2.so.7()(64bit) is needed by gimp-2:2.8.22-1.el7.x86_64
```

Cabe ao usuário encontrar os pacotes .rpm com as dependências correspondentes e instalá-los. Os gerenciadores de pacotes como o yum, zypper e dnf têm ferramentas que informam qual pacote fornece um arquivo específico. Falaremos deles mais adiante nesta lição.

Listando os pacotes instalados

Para obter uma lista de todos os pacotes instalados em seu sistema, use `rpm -qa` (o nome vem de “query all”).

```
# rpm -qa
selinux-policy-3.13.1-229.el7.noarch
pciutils-libs-3.5.1-3.el7.x86_64
redhat-menus-12.0.2-8.el7.noarch
grubby-8.28-25.el7.x86_64
hunspell-en-0.20121024-6.el7.noarch
dejavu-fonts-common-2.33-6.el7.noarch
xorg-x11-drv-dummy-0.3.7-1.el7.1.x86_64
libevdev-1.5.6-1.el7.x86_64
[...]
```

Obtendo informações sobre um pacote

Para obter informações sobre um pacote *instalado*, como o número de versão, arquitetura, data de instalação, empacotador, resumo etc., use `rpm` com os parâmetros `-qi` (o nome vem de “query info”), seguido pelo nome do pacote. Por exemplo:

```
# rpm -qi unzip
Name        : unzip
Version     : 6.0
Release     : 19.el7
Architecture: x86_64
Install Date: Sun 25 Aug 2019 05:14:39 PM EDT
Group       : Applications/Archiving
```

```

Size      : 373986
License   : BSD
Signature  : RSA/SHA256, Wed 25 Apr 2018 07:50:02 AM EDT, Key ID 24c6a8a7f4a80eb5
Source RPM : unzip-6.0-19.el7.src.rpm
Build Date : Wed 11 Apr 2018 01:24:53 AM EDT
Build Host : x86-01.bsys.centos.org
Relocations : (not relocatable)
Packager   : CentOS BuildSystem <http://bugs.centos.org>
Vendor     : CentOS
URL        : http://www.info-zip.org/UnZip.html
Summary    : A utility for unpacking zip files
Description :
The unzip utility is used to list, test, or extract files from a zip
archive. Zip archives are commonly found on MS-DOS systems. The zip
utility, included in the zip package, creates zip archives. Zip and
unzip are both compatible with archives created by PKWARE(R)'s PKZIP
for MS-DOS, but the programs' options and default behaviors do differ
in some respects.

```

Instale o pacote `unzip` se precisar listar, testar ou extrair arquivos de um arquivo `zip`.

Para ver uma lista dos arquivos que estão dentro de um pacote *instalado*, use os parâmetros `-ql` (ou seja, “query list”) seguido pelo nome do pacote:

```
# rpm -ql unzip
/usr/bin/funzip
/usr/bin/unzip
/usr/bin/unzipsfx
/usr/bin/zipgrep
/usr/bin/zipinfo
/usr/share/doc/unzip-6.0
/usr/share/doc/unzip-6.0/BUGS
/usr/share/doc/unzip-6.0/LICENSE
/usr/share/doc/unzip-6.0/README
/usr/share/man/man1/funzip.1.gz
/usr/share/man/man1/unzip.1.gz
/usr/share/man/man1/unzipsfx.1.gz
/usr/share/man/man1/zipgrep.1.gz
/usr/share/man/man1/zipinfo.1.gz
```

Se quiser informações ou uma lista de arquivos de um pacote que ainda *não* foi instalado, basta adicionar o parâmetro `-p` aos comandos acima, seguido pelo nome do arquivo RPM (FILENAME).

Assim, `rpm -qi PACKAGE`NAME se torna `rpm -qip FILENAME`, e `rpm -ql PACKAGE`NAME se torna `rpm -qlp FILENAME`, como mostrado abaixo.

```
# rpm -qip atom.x86_64.rpm
Name        : atom
Version     : 1.40.0
Release     : 0.1
Architecture: x86_64
Install Date: (not installed)
Group       : Unspecified
Size        : 570783704
License     : MIT
Signature   : (none)
Source RPM  : atom-1.40.0-0.1.src.rpm
Build Date  : sex 09 ago 2019 12:36:31 -03
Build Host  : b01bbeaf3a88
Relocations : /usr
URL         : https://atom.io/
Summary     : A hackable text editor for the 21st Century.
Description :
A hackable text editor for the 21st Century.
```

```
# rpm -qlp atom.x86_64.rpm
/usr/bin/apm
/usr/bin/atom
/usr/share/applications/atom.desktop
/usr/share/atom
/usr/share/atom/LICENSE
/usr/share/atom/LICENSES.chromium.html
/usr/share/atom/atom
/usr/share/atom/atom.png
/usr/share/atom/blink_image_resources_200_percent.pak
/usr/share/atom/content_resources_200_percent.pak
/usr/share/atom/content_shell.pak

(listing goes on)
```

Descobrindo qual pacote possui um arquivo específico

Para descobrir qual pacote instalado possui um arquivo, use o `-qf` (ou seja, “query file”) seguido pelo caminho completo para o arquivo:

```
# rpm -qf /usr/bin/unzip
unzip-6.0-19.el7.x86_64
```

No exemplo acima, o arquivo `/usr/bin/unzip` pertence ao pacote `unzip-6.0-19.el7.x86_64`.

YellowDog Updater Modified (YUM)

O `yum` foi originalmente desenvolvido como *Yellow Dog Updater* (YUP), uma ferramenta para gerenciamento de pacotes na distribuição Linux Yellow Dog. Com o tempo, evoluiu para gerenciar pacotes em outros sistemas baseados em RPM, como Fedora, CentOS, Red Hat Enterprise Linux e Oracle Linux.

Em termos de funcionalidade, ele é semelhante ao utilitário `apt` dos sistemas baseados em Debian, sendo capaz de buscar, instalar, atualizar e remover pacotes, além de gerir automaticamente as dependências. O `yum` pode ser usado para instalar um único pacote ou para atualizar um sistema inteiro de uma vez só.

Buscando pacotes

Para instalar um pacote, temos de saber o nome dele. Para isso, podemos realizar uma pesquisa com `yum search PATTERN`, onde `PATTERN` é o nome do pacote que se está procurando. O resultado é uma lista de pacotes cujo nome ou resumo contém o padrão de pesquisa especificado. Por exemplo, se você precisar de um utilitário para lidar com arquivos compactados 7Zip (com a extensão `.7z`), poderia usar:

```
# yum search 7zip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globocom
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
=====
 N/S matchyutr54ed: 7zip =====
 p7zip-plugins.x86_64 : Additional plugins for p7zip
 p7zip.x86_64 : Very high compression ratio file archiver
 p7zip-doc.noarch : Manual documentation and contrib directory
 p7zip-gui.x86_64 : 7zG – 7-Zip GUI version

Name and summary matches only, use "search all" for everything.
```

Instalando, atualizando e removendo pacotes

Para instalar um pacote usando o `yum`, use o comando `yum install PACKAGENAME`, onde `PACKAGENAME` é o nome do pacote. O `yum` busca o pacote e as dependências correspondentes em um repositório online e instala tudo no sistema.

```
# yum install p7zip
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globocom
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
Resolving Dependencies
--> Running transaction check
---> Package p7zip.x86_64 0:16.02-10.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch         Version          Repository      Size
=====
Installing:
p7zip            x86_64       16.02-10.el7    epel           604 k

Transaction Summary
=====
Install 1 Package

Total download size: 604 k
Installed size: 1.7 M
Is this ok [y/d/N]:
```

Para atualizar um pacote instalado, use `yum update PACKAGENAME`, onde `PACKAGENAME` é o nome do pacote que se deseja atualizar. Por exemplo:

```
# yum update wget
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globocom
 * extras: mirror.ufscar.br
```

```
* updates: mirror.ufscar.br
Resolving Dependencies
--> Running transaction check
---> Package wget.x86_64 0:1.14-18.el7 will be updated
---> Package wget.x86_64 0:1.14-18.el7_6.1 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package      Arch      Version      Repository      Size
=====
Updating:
wget        x86_64    1.14-18.el7_6.1    updates       547 k

Transaction Summary
=====
Upgrade 1 Package

Total download size: 547 k
Is this ok [y/d/N]:
```

Se você omitir o nome de um pacote, poderá atualizar todos os pacotes no sistema para os quais exista uma atualização disponível.

Para verificar se há uma atualização disponível para um pacote específico, use `yum check-update PACKAGE_NAME`. Como antes, se você omitir o nome do pacote, o `yum` procurará atualizações para todos os pacotes instalados no sistema.

Para remover um pacote instalado, use `yum remove PACKAGE_NAME`, onde `PACKAGE_NAME` é o nome do pacote que se deseja remover.

Como encontrar o pacote que fornece um arquivo específico

Em um exemplo anterior, mostramos uma tentativa de instalar o editor de imagens `gimp`, que falhou devido a dependências não atendidas. No entanto, o `rpm` mostra quais arquivos estão faltando, mas não lista o nome dos pacotes que os fornecem.

Por exemplo, uma das dependências ausentes era `libgimpui-2.0.so.0`. Para ver qual pacote o fornece, usamos `yum whatprovides`, seguido pelo nome do arquivo que estamos procurando:

```
# yum whatprovides libgimpui-2.0.so.0
Loaded plugins: fastestmirror, langpacks
```

```

Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
2:gimp-libs-2.8.22-1.el7.i686 : GIMP libraries
Repo      : base
Matched from:
Provides   : libgimpui-2.0.so.0

```

A resposta é `gimp-libs-2.8.22-1.el7.i686`. Em seguida podemos instalar o pacote com o comando `yum install gimp-libs`.

Isso também funciona para arquivos já existentes no seu sistema. Por exemplo, para saber de onde o arquivo `/etc/hosts` veio, você pode usar:

```

# yum whatprovides /etc/hosts
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globo.com
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
setup-2.8.71-10.el7.noarch : A set of system configuration and setup files
Repo      : base
Matched from:
Filename  : /etc/hosts

```

A resposta é `setup-2.8.71-10.el7.noarch`.

Obtendo informações sobre um pacote

Para obter informações sobre um pacote, como versão, arquitetura, descrição, tamanho e mais, use `yum info PACKAGENAME`, onde `PACKAGENAME` é o nome do pacote sobre o qual você deseja informações:

```

# yum info firefox
Last metadata expiration check: 0:24:16 ago on Sat 21 Sep 2019 02:39:43 PM -03.
Installed Packages
Name        : firefox
Version     : 69.0.1
Release    : 3.fc30
Architecture: x86_64

```

```

Size      : 268 M
Source    : firefox-69.0.1-3.fc30.src.rpm
Repository : @System
From repo  : updates
Summary    : Mozilla Firefox Web browser
URL        : https://www.mozilla.org/firefox/
License    : MPLv1.1 or GPLv2+ or LGPLv2+
Description : Mozilla Firefox is an open-source web browser, designed
              : for standards compliance, performance and portability.

```

Administrando os repositórios de software

Para o yum, os “repos” estão listados no diretório /etc/yum.repos.d/. Cada repositório é representado por um arquivo .repo, como CentOS-Base.repo.

Repositórios adicionais podem ser incluídos pelo usuário acrescentando um arquivo .repo no diretório mencionado acima, ou no final de /etc/yum.conf. No entanto, a maneira recomendada de adicionar ou gerenciar repositórios é usar a ferramenta yum-config-manager.

Para adicionar um repositório, use o parâmetro --add-repo, seguido da URL para um arquivo .repo.

```
# yum-config-manager --add-repo https://rpms.remirepo.net/enterprise/remi.repo
Loaded plugins: fastestmirror, langpacks
adding repo from: https://rpms.remirepo.net/enterprise/remi.repo
grabbing file https://rpms.remirepo.net/enterprise/remi.repo to
/etc/yum.repos.d/remi.repo
repo saved to /etc/yum.repos.d/remi.repo
```

Para obter uma lista de todos os repositórios disponíveis, use yum repolist all. A saída obtida será semelhante a esta:

```
# yum repolist all
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.ufscar.br
 * epel: mirror.globocom
 * extras: mirror.ufscar.br
 * updates: mirror.ufscar.br
repo id                  repo name          status
updates/7/x86_64          CentOS-7 - Updates   enabled: 2,500
updates-source/7           CentOS-7 - Updates Sources  disabled
```

Os repositórios `disabled` (desabilitados) serão ignorados ao instalar ou atualizar o software. Para habilitar ou desabilitar um repositório, use o utilitário `yum-config-manager`, seguido pelo ID do repositório.

Na saída acima, a identidade do repositório é mostrada na primeira coluna (`repo id`) de cada linha. Usamos apenas a parte anterior ao primeiro `/`, de forma que o id do repositório CentOS-7 – Updates é `updates`, e não `updates/7/x86_64`.

```
# yum-config-manager --disable updates
```

O comando acima desabilita o repositório `updates`. Para reativá-lo, use:

```
# yum-config-manager --enable updates
```

NOTE

O Yum armazena os pacotes baixados e os metadados associados em um diretório de cache (geralmente `/var/cache/yum`). À medida que o sistema é atualizado e novos pacotes são instalados, essa cache pode ficar bem grande. Para limpar a cache e recuperar o espaço em disco, podemos usar o comando `yum clean`, seguido pelo que deve ser removido. Os parâmetros mais úteis são `packages` (`yum clean packages`) para excluir pacotes baixados e `metadata` (`yum clean metadata`) para excluir os metadados associados. Consulte a página de manual do `yum` (digite `man yum`) para obter mais informações.

DNF

O `dnf`, a ferramenta de gerenciamento de pacotes usada no Fedora, é um fork do `yum`. Como tal, muitos dos comandos e parâmetros são semelhantes. Esta seção oferece apenas uma visão geral rápida do `dnf`.

Busca de pacotes

`dnf search PATTERN`, onde `PATTERN` é aquilo que você está buscando. Por exemplo, `dnf search unzip` mostra todos os pacotes que contêm a palavra `unzip` no nome ou descrição.

Obter informações sobre um pacote

`dnf info PACKAGENAME`

Instalar pacotes

`dnf install PACKAGENAME`, onde `PACKAGENAME` é o nome do pacote que se deseja instalar. Para encontrar o nome, faça uma busca.

Remover pacotes

```
dnf remove PACKAGENAME
```

Atualizar pacotes

`dnf upgrade` `PACKAGENAME` para atualizar um só pacote. Omita o nome do pacote para atualizar todos os pacotes do sistema.

Descobrir qual pacote fornece um arquivo específico

```
dnf provides FILENAME
```

Obter uma lista de todos os pacotes instalados no sistema

```
dnf list --installed
```

Listar o conteúdo de um pacote

```
dnf repoquery -l PACKAGENAME
```

NOTE

O `dnf` tem um sistema de ajuda embutido que mostra mais informações (como parâmetros extras) para cada comando. Para usá-lo, digite `dnf help` seguido pelo comando, como `dnf help install`.

Administrando os repositórios de software

Como no caso do `yum` e do `zypper`, o `dnf` trabalha com repositórios de software (repos). Cada distribuição tem uma lista de repositórios padrão e os administradores podem adicionar ou remover repositórios conforme necessário.

Para obter uma lista de todos os repositórios disponíveis, use `dnf repolist`. Para listar apenas os repositórios ativados, adicione a opção `--enabled` e, para listar apenas os repositórios desativados, adicione a opção `--disabled`.

```
# dnf repolist
Last metadata expiration check: 0:20:09 ago on Sat 21 Sep 2019 02:39:43 PM -03.
repo id          repo name           status
*fedora          Fedora 30 - x86_64      56,582
*fedora-modular Fedora Modular 30 - x86_64 135
*updates         Fedora 30 - x86_64 - Updates 12,774
*updates-modular Fedora Modular 30 - x86_64 - Updates 145
```

Para adicionar um repositório, use `dnf config-manager --add_repo URL`, onde `URL` é a URL completa do repositório. Para habilitar um repositório, use `dnf config-manager --set-enabled REPO_ID`.

Da mesma forma, para desativar um repositório, use `dnf config-manager --set-disabled REPO_ID`. Nos dois casos, `REPO_ID` é o ID exclusivo do repositório, que pode ser obtido com `dnf repolist`. Os repositórios adicionados são ativados por padrão. Os repositórios são armazenados em arquivos `.repo` no diretório `/etc/yum.repos.d/`, com exatamente a mesma sintaxe usada para o `yum`.

Zypper

O `zypper` é a ferramenta de gerenciamento de pacotes usada no SUSE Linux e OpenSUSE. Em termos de recursos, é semelhante ao `apt` e ao `yum`, sendo capaz de instalar, atualizar e remover pacotes de um sistema, com resolução automática de dependências.

Atualizando o índice do pacote

A exemplo de outras ferramentas de gerenciamento de pacotes, o `zypper` trabalha com repositórios que contêm pacotes e metadados. Esses metadados precisam ser atualizados periodicamente para que o utilitário fique a par dos pacotes mais recentes disponíveis. Para fazer uma atualização, basta digitar:

```
# zypper refresh
Repository 'Non-OSS Repository' is up to date.
Repository 'Main Repository' is up to date.
Repository 'Main Update Repository' is up to date.
Repository 'Update Repository (Non-Oss)' is up to date.
All repositories have been refreshed.
```

O `zypper` possui um recurso de atualização automática que pode ser ativado caso a caso, ou seja, alguns repositórios poderão ser atualizados automaticamente antes de uma consulta ou da instalação de um pacote, e outros poderão precisar ser atualizados manualmente. Vamos ensinar a controlar esse recurso em breve.

Buscando pacotes

Para procurar um pacote, use o operador `search` (ou `se`), seguido pelo nome do pacote:

```
# zypper se gnumeric
Loading repository data...
Reading installed packages...

S | Name           | Summary                                         | Type
+-----+-----+
| gnumeric        | Spreadsheet Application                         | package
```

gnumeric-devel	Spreadsheet Application	package
gnumeric-doc	Documentation files for Gnumeric	package
gnumeric-lang	Translations for package gnumeric	package

O operador de pesquisa também serve para obter uma lista de todos os pacotes instalados no sistema. Para isso, use o parâmetro `-i` sem o nome do pacote, como em `zypper se -i`.

Para ver se um pacote específico está instalado, adicione o nome do pacote ao comando acima. Por exemplo, o seguinte comando buscará dentre os pacotes instalados por um que contenha “firefox” no nome:

```
# zypper se -i firefox
Loading repository data...
Reading installed packages...

S | Name | Summary | Type
+---+-----+-----+
i | MozillaFirefox | Mozilla Firefox Web B-> | package
i | MozillaFirefox-branding-openSUSE | openSUSE branding of -> | package
i | MozillaFirefox-translations-common | Common translations f-> | package
```

Para pesquisar apenas nos pacotes *não-instalados*, adicione o parâmetro `-u` ao operador `se`.

Instalando, atualizando e removendo pacotes

Para instalar um pacote de software, use o operador `install` (ou `in`), seguido pelo nome do pacote. Desta maneira:

```
# zypper in unrar
zypper in unrar
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  unrar

1 new package to install.
Overall download size: 141.2 KiB. Already cached: 0 B. After the operation,
additional 301.6 KiB will be used.
Continue? [y/n/v/...? shows all options] (y): y
Retrieving package unrar-5.7.5-lp151.1.1.x86_64
```

```
(1/1), 141.2 KiB (301.6 KiB unpacked)
Retrieving: unrar-5.7.5-lp151.1.1.x86_64.rpm ..... [done]
Checking for file conflicts: ..... [done]
(1/1) Installing: unrar-5.7.5-lp151.1.1.x86_64 ..... [done]
```

O `zypper` também pode ser usado para instalar um pacote RPM no disco enquanto tenta atender às dependências usando pacotes dos repositórios. Para isso, basta fornecer o caminho completo para o pacote em vez de apenas um nome de pacote, como em `zypper in /home/john/newpackage.rpm`.

Para atualizar os pacotes instalados no sistema, use `zypper update`. Como no caso do processo de instalação, aparece uma lista de pacotes a serem instalados/atualizados e o gerenciador pergunta se você deseja continuar.

Se quiser listar apenas as atualizações disponíveis, sem instalar nada, use `zypper list-updates`.

Para remover um pacote, use o operador `remove` (ou `rm`), seguido pelo nome do pacote:

```
# zypper rm unrar
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following package is going to be REMOVED:
  unrar

1 package to remove.
After the operation, 301.6 KiB will be freed.
Continue? [y/n/v/...? shows all options] (y): y
(1/1) Removing unrar-5.7.5-lp151.1.1.x86_64 ..... [done]
```

Lembre-se de que a remoção de um pacote também remove outros pacotes que dependem dele. Por exemplo:

```
# zypper rm libgimp-2_0_0
Loading repository data...
Warning: No repositories defined. Operating only with the installed resolvables.
Nothing can be installed.
Reading installed packages...
Resolving package dependencies...

The following 6 packages are going to be REMOVED:
  gimp gimp-help gimp-lang gimp-plugins-python libgimp-2_0_0
```

```
libgimpui-2_0-0
```

6 packages to remove.
After the operation, 98.0 MiB will be freed.
Continue? [y/n/v/...? shows all options] (y):

Descobrindo quais pacotes possuem um arquivo específico

Para ver quais pacotes contêm um arquivo específico, use o operador de pesquisa seguido pelo parâmetro `--provides` e o nome do arquivo (ou o caminho completo para ele). Por exemplo, para saber quais pacotes contêm o arquivo `libgimpmodule-2.0.so.0` em `/usr/lib64`, usariammos:

```
# zypper se --provides /usr/lib64/libgimpmodule-2.0.so.0
Loading repository data...
Reading installed packages...

S | Name           | Summary                                         | Type
---+---+---+
i | libgimp-2_0-0 | The GNU Image Manipulation Program – Libra-> | package
```

Obtendo informações sobre um pacote

Para ver os metadados associados a um pacote, use o operador `info` seguido pelo nome do pacote. Serão exibidos o repositório de origem, nome do pacote, versão, arquitetura, fornecedor, tamanho instalado, se está instalado ou não, status (se está atualizado), o pacote de origem e uma descrição.

```
# zypper info gimp
Loading repository data...
Reading installed packages...

Information for package gimp:
-----
Repository      : Main Repository
Name            : gimp
Version         : 2.8.22-lp151.4.6
Arch            : x86_64
Vendor          : openSUSE
Installed Size  : 29.1 MiB
Installed       : Yes (automatically)
Status          : up-to-date
Source package   : gimp-2.8.22-lp151.4.6.src
Summary         : The GNU Image Manipulation Program
```

Description :

The GIMP is an image composition and editing program, which can be used for creating logos and other graphics for Web pages. The GIMP offers many tools and filters, and provides a large image manipulation toolbox, including channel operations and layers, effects, subpixel imaging and antialiasing, and conversions, together with multilevel undo. The GIMP offers a scripting facility, but many of the included scripts rely on fonts that we cannot distribute.

Administrando os repositórios de software

O `zypper` também pode ser usado para gerenciar repositórios de software. Para ver uma lista de todos os repositórios atualmente registrados no seu sistema, use `zypper repos`:

```
# zypper repos
```

Repository priorities are without effect. All enabled repositories share the same priority.

#	Alias	Name	Enabled	GPG
	Check Refresh			
1	openSUSE-Leap-15.1-1	openSUSE-Leap-15.1-1	No	
2	repo-debug	Debug Repository	No	
3	repo-debug-non-oss	Debug Repository (Non-OSS)	No	
4	repo-debug-update	Update Repository (Debug)	No	
5	repo-debug-update-non-oss	Update Repository (Debug, Non-OSS)	No	
6	repo-non-oss	Non-OSS Repository	Yes	(r)
) Yes Yes				
7	repo-oss	Main Repository	Yes	(r)
) Yes Yes				
8	repo-source	Source Repository	No	
9	repo-source-non-oss	Source Repository (Non-OSS)	No	
10	repo-update	Main Update Repository	Yes	(r)
) Yes Yes				
11	repo-update-non-oss	Update Repository (Non-Oss)	Yes	(r)

) Yes | Yes

Na coluna Enabled vemos que alguns repositórios estão ativados e outros não. Para mudar isso, use o operador modifyrepo, seguido pelo parâmetro -e (enable) ou -d (disable) e pelo alias do repositório (a segunda coluna na saída acima).

```
# zypper modifyrepo -d repo-non-oss
Repository 'repo-non-oss' has been successfully disabled.

# zypper modifyrepo -e repo-non-oss
Repository 'repo-non-oss' has been successfully enabled.
```

Mencionamos anteriormente que o zypper possui um recurso de *atualização automática* que pode ser ativado caso a caso nos repositórios. Quando ativado, esse sinalizador faz com que o zypper execute uma operação de atualização (como se executássemos zypper refresh) antes de trabalhar com o repositório especificado. O processo pode ser controlado com os parâmetros -f e -F do operador modifyrepo:

```
# zypper modifyrepo -F repo-non-oss
Autorefresh has been disabled for repository 'repo-non-oss'.

# zypper modifyrepo -f repo-non-oss
Autorefresh has been enabled for repository 'repo-non-oss'.
```

Adicionando e removendo repositórios

Para adicionar um novo repositório de software para o zypper, use o operador addrepo seguido da URL e do nome do repositório, como abaixo:

```
# zypper addrepo http://packman.inode.at/suse/openSUSE_Leap_15.1/ packman
Adding repository 'packman' ..... [done]
Repository 'packman' successfully added

URI      : http://packman.inode.at/suse/openSUSE_Leap_15.1/
Enabled   : Yes
GPG Check : Yes
Autorefresh : No
Priority   : 99 (default priority)
```

Repository priorities are without effect. All enabled repositories share the same

priority.

Ao adicionar um repositório, podemos ativar as atualizações automáticas com o parâmetro `-f`. Os repositórios adicionados são ativados por padrão, mas também é possível adicionar e desativar um repositório ao mesmo tempo usando o parâmetro `-d`. Para remover um repositório, use o operador `removerrepo` seguido pelo nome do repositório (Alias). Para remover o repositório adicionado no exemplo acima, o comando seria:

```
# zypper removerrepo packman
Removing repository 'packman' ..... [done]
Repository 'packman' has been removed.
```

Exercícios Guiados

1. Usando o `rpm` em um sistema Red Hat Enterprise Linux, como você instalaria o pacote `file-roller-3.28.1-2.el7.x86_64.rpm` de maneira a exibir uma barra de progresso durante a instalação?

2. Usando o `rpm`, descubra qual pacote contém o arquivo `/etc/redhat-release`.

3. Como você usaria o `yum` para procurar atualizações para todos os pacotes do sistema?

4. Usando o `zypper`, como desabilitaríamos um repositório chamado `repo-extras`?

5. Se tivermos um arquivo `.repo` descrevendo um novo repositório, onde esse arquivo deve ser colocado para poder ser reconhecido pelo DNF?

Exercícios Exploratórios

1. Como usar o `zypper` para descobrir qual pacote possui o arquivo `/usr/sbin/swapon`?

2. Como obter uma lista de todos os pacotes instalados no sistema usando o `dnf`?

3. Usando o `dnf`, qual o comando para adicionar um repositório localizado em `https://www.example.url/home:reponame.repo` ao sistema?

4. Como podemos usar o `zypper` para conferir se o pacote `unzip` está instalado?

5. Usando o `yum`, descubra qual pacote fornece o arquivo `/bin/wget`.

Resumo

Nesta lição, você aprendeu:

- Como usar o `rpm` para instalar, atualizar e remover pacotes.
- Como usar o `yum`, o `zypper` e o `dnf`.
- Como obter informações sobre um pacote.
- Como obter uma lista do conteúdo do pacote.
- Como descobrir de qual pacote veio um arquivo.
- Como listar, adicionar, remover, habilitar ou desabilitar repositórios de software.

Os seguintes comandos foram abordados:

- `rpm`
- `yum`
- `dnf`
- `zypper`

Respostas aos Exercícios Guiados

1. Usando o `rpm` em um sistema Red Hat Enterprise Linux, como você instalaria o pacote `file-roller-3.28.1-2.el7.x86_64.rpm` de maneira a exibir uma barra de progresso durante a instalação?

Use o parâmetro `-i` para instalar um pacote e a opção `-h` para habilitar os “sinais de hash” mostrando o progresso da instalação. Assim, a resposta é: `rpm -ih file-roller-3.28.1-2.el7.x86_64.rpm`.

2. Usando o `rpm`, descubra qual pacote contém o arquivo `/etc/redhat-release`.

Estamos solicitando informações sobre um arquivo, por isso usamos o parâmetro `-qf`: `rpm -qf /etc/redhat-release`.

3. Como você usaria o `yum` para procurar atualizações para todos os pacotes do sistema?

Use a operação `check-update` sem um nome de pacote: `yum check-update`.

4. Usando o `zypper`, como desabilitaríamos um repositório chamado `repo-extras`?

Use a operação `modifyrepo` para alterar os parâmetros de um repositório, e o parâmetro `-d` para desabilitá-lo: `zypper modifyrepo -d repo-extras`.

5. Se tivermos um arquivo `.repo` descrevendo um novo repositório, onde esse arquivo deve ser colocado para poder ser reconhecido pelo DNF?

Os arquivos `.repo` para o DNF devem ser postos no mesmo local usado pelo YUM, dentro de `/etc/yum.repos.d/`.

Respostas aos Exercícios Exploratórios

1. Como usar o `zypper` para descobrir qual pacote possui o arquivo `/usr/sbin/swapon`?

Use o operador `se` (`search`) e o parâmetro `--provides`: `zypper se --provides /usr/sbin/swapon`.

2. Como obter uma lista de todos os pacotes instalados no sistema usando o `dnf`?

Use o operador `list`, seguido pelo parâmetro `--installed`: `dnf list --installed`.

3. Usando o `dnf`, qual o comando para adicionar um repositório localizado em `https://www.example.url/home:reponame.repo` ao sistema?

O trabalho com repositórios é uma “mudança de configurações”, por isso use o `config-manager` e o parâmetro `--add_repo`: `dnf config-manager --add_repo https://www.example.url/home:reponame.repo`.

4. Como podemos usar o `zypper` para conferir se o pacote `unzip` está instalado?

Precisamos fazer uma busca (`se`) nos pacotes instalados (`-i`): `zypper se -i unzip`.

5. Usando o `yum`, descubra qual pacote fornece o arquivo `/bin/wget`.

Para descobrir quem fornece um arquivo, use `whatprovides` e o nome do arquivo: `yum whatprovides /bin/wget`.



102.6 Linux virtualizado

Referência ao LPI objectivo

[LPIC-1, Exam 101, Objective 102.6](#)

Peso

1

Áreas chave de conhecimento

- Entender o conceito geral de máquinas virtuais e contêineres.
- Entender elementos comuns em máquinas virtuais numa nuvem IaaS, como instâncias computacionais, armazenamento em bloco e rede.
- Entender as propriedades exclusivas de um sistema Linux que precisam ser alteradas quando um sistema é clonado ou utilizado como modelo.
- Entender como imagens de sistema são utilizadas para implementar máquinas virtuais, instâncias de nuvem e contêineres.
- Entender as extensões do Linux que integram o Linux com uma solução de virtualização.
- Noções de cloud-init.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- Máquina Virtual
- Contêiner Linux
- Contêiner de Aplicação
- Drivers de convidado
- Chaves SSH do host
- Id de máquina D-Bus



**Linux
Professional
Institute**

102.6 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	102 Instalação do Linux e gerenciamento de pacotes
Objetivo:	102.6 O Linux como máquina virtual
Lição:	1 de 1

Introdução

Um dos maiores pontos fortes do Linux é sua versatilidade. Um aspecto dessa versatilidade é a possibilidade de usar o Linux como meio de hospedar outros sistemas operacionais, ou aplicativos individuais, em um ambiente completamente isolado e seguro. Esta lição será dedicada aos conceitos de virtualização e tecnologias de contêiner, juntamente com alguns detalhes técnicos que devem ser levados em conta ao se implementar uma máquina virtual em uma plataforma de nuvem.

O que é virtualização?

A virtualização é uma tecnologia que permite que uma plataforma de software, chamada de *hipervisor*, execute processos que contêm um sistema inteiramente emulado (virtual). O hipervisor é responsável por gerenciar os recursos do hardware físico que podem ser usados por máquinas virtuais individuais. Essas máquinas virtuais são chamadas de *guests* (convidados) do hypervisor. Muitos dos aspectos de um computador físico são emulados por software na máquina virtual, como a BIOS do sistema e os controladores de disco rígido. Uma máquina virtual geralmente usa imagens de disco rígido que são armazenadas como arquivos individuais e tem acesso à RAM e à CPU da máquina hospedeira por meio do software hipervisor. O hipervisor divide o acesso aos recursos de

hardware do sistema hospedeiro entre os convidados, permitindo assim que vários sistemas operacionais sejam executados em um único sistema hospedeiro.

Dentre os hipervisores mais comumente usados no Linux, podemos citar:

Xen

O Xen é um hipervisor de código aberto de Tipo 1, o que significa que ele não depende de um sistema operacional subjacente para funcionar. Um hipervisor desse tipo é conhecido como *hipervisor bare-metal*, pois o computador pode inicializar diretamente no hipervisor.

KVM

O Kernel Virtual Machine é um módulo de virtualização do kernel do Linux. O KVM é um hipervisor de Tipo 1 e também de Tipo 2 porque, embora precise de um sistema operacional Linux genérico para funcionar, é capaz de agir perfeitamente bem como hipervisor integrando-se a uma instalação Linux em execução. As máquinas virtuais implementadas com o KVM usam o daemon `libvirt` e utilitários de software associados para serem criadas e gerenciadas.

VirtualBox

Um aplicativo desktop popular que facilita a criação e o gerenciamento de máquinas virtuais. O Oracle VM VirtualBox é multiplataforma e funciona em Linux, macOS e Microsoft Windows. Como o VirtualBox requer um sistema operacional subjacente para ser executado, ele é um hipervisor de Tipo 2.

Alguns hipervisores permitem a realocação dinâmica de uma máquina virtual. O processo de mover uma máquina virtual de uma instalação do hipervisor para outra é chamado de *migração* e as técnicas envolvidas são diferentes conforme as implementações do hipervisor. Algumas migrações só podem ser realizadas quando o sistema convidado está completamente desligado e outras podem ser feitas com o convidado em execução (é o que chamamos *migração ao vivo*). Essas técnicas podem ser úteis durante a manutenção dos hipervisores, ou ainda para garantir a resiliência do sistema quando um hipervisor para de funcionar, sendo possível mover o convidado para outro.

Tipos de máquina virtual

Existem três tipos principais de máquinas virtuais, o convidado *totalmente virtualizado*, o convidado *paravirtualizado* e o convidado *híbrido*.

Totalmente virtualizado

Todas as instruções que um sistema operacional convidado precisa executar devem poder rodar em uma instalação de sistema operacional totalmente virtualizada. A razão para isso é que nenhum driver de software adicional é instalado na máquina virtual para traduzir as instruções para o hardware simulado ou real. O convidado totalmente virtualizado (ou HardwareVM) não

sabe que é uma instância de máquina virtual em execução. Para que esse tipo de virtualização seja possível com hardware baseado em x86, as extensões de CPU Intel VT-x ou AMD-V devem ser habilitadas no sistema no qual o hipervisor está instalado. Isso pode ser feito a partir de um menu de configuração de firmware na BIOS ou UEFI.

Paravirtualizado

Um convidado paravirtualizado (ou PVM) é aquele que está ciente de ser uma instância de máquina virtual em execução. Esses tipos de convidados fazem uso de um kernel modificado e drivers especiais (conhecidos como *drivers convidados*) que ajudam o sistema operacional convidado a utilizar os recursos de software e hardware do hipervisor. O desempenho de um convidado paravirtualizado costuma ser melhor do que o de um convidado totalmente virtualizado devido à vantagem oferecida por esses drivers de software.

Híbrido

A paravirtualização e a virtualização total podem ser combinadas para permitir que sistemas operacionais não modificados tenham um desempenho de E/S quase nativo usando drivers paravirtualizados em sistemas operacionais totalmente virtualizados. Os drivers paravirtualizados contêm drivers de dispositivos de armazenamento e de rede com desempenho aprimorado de E/S de disco e de rede.

As plataformas de virtualização geralmente fornecem drivers convidados para os sistemas operacionais virtualizados. O KVM utiliza drivers do projeto *Virtio*, enquanto o Oracle VM VirtualBox usa *Extensões de Convidado* disponíveis em um arquivo de imagem ISO de CD-ROM para download.

Exemplo de máquina virtual libvirt

Veremos um exemplo de máquina virtual que é gerenciada por `libvirt` e usa o hipervisor KVM. Uma máquina virtual geralmente consiste em um grupo de arquivos, principalmente um arquivo XML que *define* a máquina virtual (sua configuração de hardware, conectividade de rede, recursos de exibição etc.) e um arquivo de imagem de disco rígido associado contendo a instalação do sistema operacional e seu software.

Primeiro, vamos examinar um exemplo de arquivo de configuração XML para uma máquina virtual e seu ambiente de rede:

```
$ ls /etc/libvirt/qemu
total 24
drwxr-xr-x 3 root root 4096 Oct 29 17:48 networks
-rw----- 1 root root 5667 Jun 29 17:17 rhel8.0.xml
```

NOTE

A parte qemu do caminho do diretório se refere ao software subjacente do qual dependem as máquinas virtuais baseadas em KVM. O projeto QEMU fornece software para o hipervisor para emular os dispositivos de hardware que serão usados pela máquina virtual, como controladores de disco, acesso à CPU do host, emulação de placa de rede e muito mais.

Observe que existe um diretório chamado networks. Esse diretório contém arquivos de definição (que também usam XML) que criam configurações de rede utilizáveis pelas máquinas virtuais. Este hipervisor está usando apenas uma rede e, portanto, há apenas um arquivo de definição contendo uma configuração para um segmento de rede virtual que será usado por esses sistemas.

```
$ ls -l /etc/libvirt/qemu/networks/
total 8
drwxr-xr-x 2 root root 4096 Jun 29 17:15 autostart
-rw----- 1 root root 576 Jun 28 16:39 default.xml
$ sudo cat /etc/libvirt/qemu/networks/default.xml
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh net-edit default
or other application using the libvirt API.
-->

<network>
  <name>default</name>
  <uuid>55ab064f-62f8-49d3-8d25-8ef36a524344</uuid>
  <forward mode='nat' />
  <bridge name='virbr0' stp='on' delay='0' />
  <mac address='52:54:00:b8:e0:15' />
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
    </dhcp>
  </ip>
</network>
```

Esta definição inclui uma rede privada Classe C e um dispositivo de hardware emulado que age como um roteador para esta rede. Também vemos um intervalo de endereços IP que o hipervisor, junto com uma implementação de servidor DHCP, pode atribuir às máquinas virtuais que usam esta rede. Essa configuração de rede também utiliza a *tradução de endereços de rede* (NAT) para encaminhar pacotes para outras redes, como a LAN do hipervisor.

Veremos a seguir um arquivo de definição de máquina virtual Red Hat Enterprise Linux 8 (as seções mais importantes estão em negrito):

```
$ sudo cat /etc/libvirt/qemu/rhel8.0.xml
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:
  virsh edit rhel8.0
or other application using the libvirt API.
-->

<domain type='kvm'>
  <name>rhel8.0</name>
  <uuid>fadd8c5d-c5e1-410e-b425-30da7598d0f6</uuid>
  <metadata>
    <libosinfo:libosinfo
      xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/8.0"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-q35-3.1'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off' />
  </features>
  <cpu mode='host-model' check='partial'>
    <model fallback='allow' />
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup' />
    <timer name='pit' tickpolicy='delay' />
    <timer name='hpet' present='no' />
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
```

```
<suspend-to-mem enabled='no' />
<suspend-to-disk enabled='no' />
</pm>
<devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
        <driver name='qemu' type='qcow2' />
        <source file='/var/lib/libvirt/images/rhel8' />
        <target dev='vda' bus='virtio' />
        <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0' />
    </disk>
    <controller type='usb' index='0' model='qemu-xhci' ports='15'>
        <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0' />
    </controller>
    <controller type='sata' index='0'>
        <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2' />
    </controller>
    <controller type='pci' index='0' model='pcie-root' />
    <controller type='pci' index='1' model='pcie-root-port'>
        <model name='pcie-root-port' />
        <target chassis='1' port='0x10' />
        <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'
multifunction='on' />
    </controller>
    <controller type='pci' index='2' model='pcie-root-port'>
        <model name='pcie-root-port' />
        <target chassis='2' port='0x11' />
        <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1' />
    </controller>
    <controller type='pci' index='3' model='pcie-root-port'>
        <model name='pcie-root-port' />
        <target chassis='3' port='0x12' />
        <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2' />
    </controller>
    <controller type='pci' index='4' model='pcie-root-port'>
        <model name='pcie-root-port' />
        <target chassis='4' port='0x13' />
        <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3' />
    </controller>
    <controller type='pci' index='5' model='pcie-root-port'>
        <model name='pcie-root-port' />
        <target chassis='5' port='0x14' />
        <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x4' />
    </controller>
    <controller type='pci' index='6' model='pcie-root-port'>
```

```

<model name='pcie-root-port'/>
<target chassis='6' port='0x15'/>
<address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x5'/>
</controller>
<controller type='pci' index='7' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='7' port='0x16'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x6'/>
</controller>
<controller type='virtio-serial' index='0'>
    <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
</controller>
<interface type='network'>
    <mac address='52:54:00:50:a7:18'/>
    <source network='default'/>
    <model type='virtio'/'>
        <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/'>
</interface>
<serial type='pty'>
    <target type='isa-serial' port='0'>
        <model name='isa-serial'/'>
    </target>
</serial>
<console type='pty'>
    <target type='serial' port='0'/'>
</console>
<channel type='unix'>
    <target type='virtio' name='org.qemu.guest_agent.0'/'>
    <address type='virtio-serial' controller='0' bus='0' port='1'/'>
</channel>
<channel type='spicevmc'>
    <target type='virtio' name='com.redhat.spice.0'/'>
    <address type='virtio-serial' controller='0' bus='0' port='2'/'>
</channel>
<input type='tablet' bus='usb'>
    <address type='usb' bus='0' port='1'/'>
</input>
<input type='mouse' bus='ps2'/'>
<input type='keyboard' bus='ps2'/'>
<graphics type='spice' autoport='yes'>
    <listen type='address'/'>
    <image compression='off'/'>
</graphics>
<sound model='ich9'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/'>

```

```

</sound>
<video>
    <model type='virtio' heads='1' primary='yes'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0' />
</video>
<redirdev bus='usb' type='spicevmc'>
    <address type='usb' bus='0' port='2' />
</redirdev>
<redirdev bus='usb' type='spicevmc'>
    <address type='usb' bus='0' port='3' />
</redirdev>
<memballoon model='virtio'>
    <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0' />
</memballoon>
<rng model='virtio'>
    <backend model='random'>/dev/urandom</backend>
    <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0' />
</rng>
</devices>
</domain>

```

Este arquivo define uma série de configurações de hardware que são usadas por este convidado, como a quantidade de RAM que lhe será atribuída, o número de núcleos da CPU do hipervisor a que o convidado terá acesso, o arquivo de imagem de disco rígido associado a este convidado (na estrofe disk), seus recursos de exibição (por meio do protocolo SPICE) e o acesso do convidado a dispositivos USB, bem como o teclado emulado e a entrada de mouse.

Exemplo de armazenamento em disco de uma máquina virtual

A imagem de disco rígido desta máquina virtual reside em `/var/lib/libvirt/images/rhel8`. Eis a imagem de disco em si neste hipervisor:

```
$ sudo ls -lh /var/lib/libvirt/images/rhel8
-rw----- 1 root root 5.5G Oct 25 15:57 /var/lib/libvirt/images/rhel8
```

O tamanho atual desta imagem de disco consome apenas 5,5 GB de espaço no hipervisor. No entanto, o sistema operacional deste convidado vê um disco de 23,3 GB, conforme evidenciado pela saída do seguinte comando lançado dentro da máquina virtual em execução:

\$ lsblk					
NAME	MAJ:MIN	RM	SIZE	TYPE	MOUNTPOINT

```
vda      252:0    0 23.3G  0 disk
└─vda1   252:1    0     1G  0 part /boot
└─vda2   252:2    0 22.3G  0 part
└─rhel-root 253:0  0    20G  0 lvm  /
└─rhel-swap 253:1  0    2.3G  0 lvm  [SWAP]
```

Isso se deve ao tipo de provisionamento de disco usado para este convidado. Existem vários tipos de imagens de disco que uma máquina virtual pode usar, mas os dois principais são:

COW

Copy-on-write ou cópia na gravação (também conhecida como *thin-provisioning* ou *sparse images*) é um método em que um arquivo de disco é criado com um limite máximo de tamanho predefinido. O tamanho da imagem do disco só aumenta quando novos dados são gravados no disco. Como no exemplo anterior, o sistema operacional convidado vê o limite de disco predefinido de 23,3 GB, mas gravou apenas 5,5 GB de dados no arquivo de disco. O formato de imagem de disco usado para a máquina virtual de exemplo é qcow2, um formato de arquivo QEMU COW.

RAW

Um tipo de disco *raw* ou *full* é um arquivo que tem todo o seu espaço pré-alocado. Por exemplo, um arquivo de imagem de disco raw de 10 GB consome 10 GB de espaço em disco real no hipervisor. Esse estilo de disco permite um ganho de desempenho, pois todo o espaço em disco necessário já existe, de modo que o hipervisor subjacente pode simplesmente gravar dados no disco sem o impacto no desempenho que seria causado pela necessidade de monitorar a imagem do disco para garantir que ele não atingiu seu limite e de estender o tamanho do arquivo à medida que novos dados são gravados nele.

Existem outras plataformas de gerenciamento de virtualização, como o *Red Hat Enterprise Virtualization* e o *oVirt*, que podem lançar mão de discos físicos como locais de armazenamento secundários para o sistema operacional de uma máquina virtual. Esses sistemas utilizam dispositivos de rede de área de armazenamento (SAN) ou de armazenamento conectado à rede (NAS) para gravar seus dados, e o hipervisor controla quais locais de armazenamento pertencem a quais máquinas virtuais. Esses sistemas de armazenamento podem usar tecnologias como o gerenciamento de volume lógico (LVM) para aumentar ou diminuir o tamanho do armazenamento em disco de uma máquina virtual conforme necessário e para auxiliar na criação e gerenciamento de instantâneos de armazenamento.

Trabalhando com modelos de máquina virtual

Como as máquinas virtuais são, tipicamente, apenas arquivos em execução em um hipervisor, é fácil criar *modelos* que podem ser personalizados para cenários específicos de implantação.

Freqüentemente, uma máquina virtual contém uma instalação básica de sistema operacional e algumas configurações de autenticação predefinidas para facilitar futuras inicializações do sistema. Isso diminui o tempo necessário para construir um novo sistema, reduzindo a quantidade de trabalho repetitivo, como a instalação de pacotes básicos e as configurações de localidade.

Esse modelo de máquina virtual pode ser copiado posteriormente para um novo sistema convidado. Nesse caso, o novo convidado receberia um novo nome e um novo endereço MAC para sua interface de rede, além de outras modificações dependendo do uso pretendido.

O D-Bus Machine ID

Muitas instalações do Linux utilizam um número de identificação de máquina gerado no momento da instalação, chamado de *D-Bus machine ID*. No entanto, se uma máquina virtual for *clonada* para ser usada como modelo para outras instalações de máquina virtual, um novo D-Bus machine ID precisará ser criado para garantir que os recursos do sistema do hipervisor sejam direcionados ao sistema convidado correto.

O comando a seguir serve para validar se existe um D-Bus machine ID para o sistema em execução:

```
$ dbus-uuidgen --ensure
```

Se nenhuma mensagem de erro for exibida, é porque já existe um ID para o sistema. Para visualizar o D-Bus machine ID atual, execute o seguinte:

```
$ dbus-uuidgen --get  
17f2e0698e844e31b12cc3f9aa4d94a
```

A sequência de caracteres exibida é o número de ID atual. Dois sistemas Linux em execução em um hipervisor não devem ter o mesmo D-Bus machine ID.

O D-Bus machine ID localiza-se em `/var/lib/dbus/machine-id` e está simbolicamente ligado a `/etc/machine-id`. Não é recomendável alterar esse número de ID em um sistema em execução, pois isso pode acarretar instabilidades e travamentos do sistema. Se duas máquinas virtuais tiverem o mesmo D-Bus machine ID, siga o procedimento abaixo para gerar um novo:

```
$ sudo rm -f /etc/machine-id  
$ sudo dbus-uuidgen --ensure=/etc/machine-id
```

Se por acaso `/var/lib/dbus/machine-id` não for um link simbólico que remete a `/etc/machine-id`,

`/var/lib/dbus/machine-id` terá de ser removido.

Implementação de máquinas virtuais na nuvem

Muitos provedores de IaaS (*infraestrutura como serviço*) executam sistemas de hipervisor e podem implantar imagens de máquinas virtuais para uma empresa. Praticamente todos esses provedores oferecem ferramentas que permitem a um administrador construir, implementar e configurar máquinas virtuais personalizadas com base em uma variedade de distribuições Linux. Muitos deles também possuem sistemas que permitem a implementação e migração de máquinas virtuais construídas de dentro da empresa de um cliente.

Ao avaliar a implementação de um sistema Linux em um ambiente IaaS, existem alguns elementos-chave que um administrador deve conhecer:

Instâncias de computação

Muitos provedores de nuvem cobram taxas de uso com base em “instâncias de computação”, ou o tempo de CPU que será usado por uma infraestrutura baseada em nuvem. O planejamento cuidadoso do tempo de processamento real exigido por cada aplicativo ajuda a manter gerenciáveis os custos de uma solução em nuvem.

As instâncias de computação também se referem ao número de máquinas virtuais provisionadas em um ambiente de nuvem. Também neste caso o maior número de instâncias de sistemas em execução ao mesmo tempo influencia no tempo geral de CPU que uma empresa terá de pagar.

Armazenamento em bloco

Os provedores de nuvem também oferecem diversos níveis de armazenamento em bloco para uso empresarial. Algumas ofertas consistem simplesmente em armazenamento de rede baseado na web para arquivos; outras oferecem armazenamento externo para uma máquina virtual provisionada em nuvem e usada para hospedar arquivos.

O custo dessas ofertas varia de acordo com a quantidade de armazenamento usada e a velocidade do armazenamento nos data centers do provedor. Um acesso mais rápido ao armazenamento normalmente custa mais e, inversamente, os dados "em repouso" (como no caso do arquivamento de dados) costumam ser bem baratos.

Rede

Um dos principais componentes do trabalho com um provedor de soluções em nuvem é a maneira como a rede virtual será configurada. Muitos provedores de IaaS oferecem alguma forma de utilitários baseados na web que podem ser utilizados para o projeto e implementação de diferentes rotas de rede, sub-redes e configurações de firewall. Alguns fornecem até mesmo soluções de DNS para ser possível atribuir FQDN (nomes de domínio absolutos) publicamente

acessíveis aos sistemas de Internet do cliente. Existem também soluções “híbridas” capazes de conectar uma infraestrutura de rede local existente a uma infraestrutura baseada em nuvem por meio de uma VPN (*rede privada virtual*), unindo assim as duas infraestruturas.

Acessando convidados na nuvem com segurança

O método mais comum para acessar uma máquina virtual remota em uma plataforma de nuvem é por meio do software OpenSSH. Um sistema Linux residente na nuvem teria o servidor OpenSSH em execução, enquanto um administrador usaria um cliente OpenSSH com chaves pré-compartilhadas para acesso remoto.

Um administrador executaria o seguinte comando:

```
§ ssh-keygen
```

segundo as instruções para criar um par de chaves SSH públicas e privadas. A chave privada permanece no sistema local do administrador (armazenada em `~/.ssh/`) e a chave pública é copiada para o sistema de nuvem remoto, exatamente o mesmo método que se usaria ao trabalhar com máquinas em rede em uma LAN corporativa.

O administrador então executaria o seguinte comando:

```
§ ssh-copy-id -i <public_key> user@cloud_server
```

Ele copia a chave SSH pública do par que acaba de ser gerado para o servidor de nuvem remoto. A chave pública é gravada no arquivo `~/.ssh/authorized_keys` do servidor na nuvem e define as permissões apropriadas no arquivo.

NOTE

Se houver apenas um arquivo de chave pública no diretório `~/.ssh/`, então a opção `-i` pode ser omitida, pois o comando `ssh-copy-id` será o padrão para o arquivo de chave pública no diretório (normalmente o arquivo que termina com a extensão `.pub`).

Alguns provedores de nuvem geram automaticamente um par de chaves quando um novo sistema Linux é provisionado. O administrador precisa então baixar a chave privada para o novo sistema do provedor de nuvem e armazená-la em seu sistema local. Observe que as permissões para chaves SSH devem ser `0600` para uma chave privada e `0644` para uma chave pública.

Pré-configurando sistemas em nuvem

Uma ferramenta útil que simplifica as implementações de máquina virtual baseada em nuvem é o utilitário `cloud-init`. Esse comando, junto com os arquivos de configuração associados e a imagem de máquina virtual predefinida, é um método neutro (independente de fornecedor) para implementar um convidado Linux em uma infinidade de provedores de IaaS. Utilizando arquivos de texto simples YAML (*YAML Ain't Markup Language*), o administrador pode pré-configurar as definições de rede, a seleção de pacotes de software, a configuração de chave SSH, a criação de contas de usuário e as configurações de localidade, além de uma série de outras opções para criar novos sistemas rapidamente.

Durante a inicialização de um novo sistema, o `cloud-init` lê as definições dos arquivos de configuração YAML e as aplica. Basta efetuar esse processo na configuração inicial de um sistema e a implantação de uma frota de novos sistemas em uma plataforma de provedor de nuvem ficará muito mais fácil.

A sintaxe do arquivo YAML usado com `cloud-init` se chama *cloud-config*. Eis um exemplo de arquivo `cloud-config`:

```
#cloud-config
timezone: Africa/Dar_es_Salaam
hostname: test-system

# Update the system when it first boots up
apt_update: true
apt_upgrade: true

# Install the Nginx web server
packages:
  - nginx
```

Observe que na linha superior não há espaço entre o símbolo hash (#) e o termo `cloud-config`.

NOTE

`cloud-init` não é apenas para máquinas virtuais. O conjunto de ferramentas `cloud-init` também pode ser usado para pré-configurar contêineres (como contêineres LXD Linux) antes da implementação.

Contêiners

A tecnologia de contêiner lembra uma máquina virtual em certos aspectos: eles proporcionam um ambiente isolado para implementar facilmente um aplicativo. Ao passo que, com uma máquina

virtual, um computador inteiro é emulado, um contêiner usa apenas o software suficiente para executar um aplicativo. Dessa forma, há muito menos sobrecarga.

Os contêineres permitem uma maior flexibilidade em relação a uma máquina virtual. Um contêiner de aplicativo pode ser migrado de um hospedeiro para outro, assim como uma máquina virtual pode ser migrada de um hipervisor para outro. No entanto, às vezes, uma máquina virtual precisa ser desligada antes de poder ser migrada, enquanto no contêiner o aplicativo permanece em execução enquanto está sendo migrado. Os contêineres também facilitam a implementação de novas versões de aplicativos em conjunto com uma versão existente. Conforme os usuários fecham suas sessões com contêineres em execução, esses contêineres podem ser removidos automaticamente do sistema pelo software de orquestração de contêineres e substituídos pela nova versão, reduzindo assim o tempo de inatividade.

NOTE Existem várias tecnologias de contêiner disponíveis para Linux, como *Docker*, *Kubernetes*, *LXD / LXC*, *systemd-nspawn*, *OpenShift* e outros. A implementação exata de um pacote de software contêiner está além do escopo do exame LPIC-1.

Os contêineres usam o mecanismo de *grupos de controle* (mais conhecido como *cgroups*) no kernel do Linux. O cgroup é uma forma de particionar os recursos do sistema, como memória, tempo do processador, bem como disco e largura de banda da rede, para um aplicativo individual. Um administrador pode usar cgroups diretamente para definir os limites de recursos do sistema em um aplicativo ou um grupo de aplicativos existentes em um único cgroup. Em essência, é isso o que o software de contêiner faz para o administrador, além de fornecer ferramentas que facilitam o gerenciamento e a implementação de cgroups.

NOTE Atualmente, o conhecimento de cgroups não é necessário para prestar o exame LPIC-1. O conceito de cgroup é mencionado aqui para que o candidato tenha pelo menos algum conhecimento prévio de como um aplicativo é segregado para fins de utilização dos recursos do sistema.

Exercícios Guiados

- Quais extensões de CPU são necessárias em uma plataforma de hardware baseada em x86 para rodar convidados totalmente virtualizados?

- Uma instalação de servidor de missão crítica, que exige um desempenho mais rápido, provavelmente usará que tipo de virtualização?

- Duas máquinas virtuais que foram clonadas a partir do mesmo modelo e que utilizam o D-Bus apresentam desempenho irregular. Ambas têm nomes de host e definições de configuração de rede separadas. Qual comando seria usado para determinar se cada uma das máquinas virtuais tem diferentes D-Bus Machine IDs?

Exercícios Exploratórios

Execute o seguinte comando para ver se seu sistema já tem extensões de CPU habilitadas para rodar uma máquina virtual (os resultados podem variar dependendo de sua CPU):

```
+ grep --color -E "vmx|svm" /proc/cpuinfo
```

+ Dependendo da saída, `vmx` pode estar realçado (para CPUs habilitadas com Intel VT-x) ou, ainda, `svm` (para CPUs habilitadas com AMD SVM). Se não obtiver resultados, consulte as instruções da BIOS ou da UEFI sobre como habilitar a virtualização para o seu processador.

+

1. Se o seu processador suporta virtualizações, procure a documentação da sua distribuição para executar um hipervisor KVM.

- Instale os pacotes necessários para executar um hipervisor KVM.

```
[REDACTED]
```

- Se estiver usando um ambiente de desktop gráfico, é recomendado instalar também o aplicativo `virt-manager`, um front-end gráfico que pode ser usado em uma instalação KVM. Isso ajudará na instalação e gerenciamento da máquina virtual.

```
[REDACTED]
```

- Baixe a imagem ISO de uma distribuição Linux à sua escolha e, seguindo a documentação da distribuição, crie uma nova máquina virtual usando essa ISO.

```
[REDACTED]
```

Resumo

Nesta lição, cobrimos os conceitos básicos de máquinas virtuais e contêineres e como essas tecnologias podem ser usadas com o Linux.

Descrevemos resumidamente os seguintes comandos:

dbus-uuidgen

Usado para verificar e visualizar a ID DBus de um sistema.

ssh-keygen

Usado para gerar um par de chaves SSH públicas e privadas para uso ao acessar sistemas remotos baseados em nuvem.

ssh-copy-id

Usado para copiar a chave SSH pública de um sistema para um sistema remoto para facilitar a autenticação remota.

cloud-init

Usado para auxiliar na configuração e implementação de máquinas virtuais e contêineres em um ambiente de nuvem.

Respostas aos Exercícios Guiados

- Quais extensões de CPU são necessárias em uma plataforma de hardware baseada em x86 para rodar convidados totalmente virtualizados?

VT-x para as CPUs Intel ou AMD-V para CPUs AMD

- Uma instalação de servidor de missão crítica, que exige um desempenho mais rápido, provavelmente usará que tipo de virtualização?

Um sistema operacional que faz uso de paravirtualização, como o Xen, pois o sistema operacional convidado poderá fazer melhor uso dos recursos de hardware disponíveis com drivers de software projetados para funcionar com o hipervisor.

- Duas máquinas virtuais que foram clonadas a partir do mesmo modelo e que utilizam o D-Bus apresentam desempenho irregular. Ambas têm nomes de host e definições de configuração de rede separadas. Qual comando seria usado para determinar se cada uma das máquinas virtuais tem diferentes D-Bus Machine IDs?

`dbus-uuidgen --get`

Respostas aos Exercícios Exploratórios

- Execute o seguinte comando para ver se seu sistema já tem extensões de CPU habilitadas para rodar uma máquina virtual (os resultados podem variar dependendo de sua CPU): `grep --color -E "vmx|svm" /proc/cpuinfo`. Dependendo da saída, `vmx` pode estar realçado (para CPUs habilitadas com Intel VT-x) ou, ainda, `svm` (para CPUs habilitadas com AMD SVM). Se não obtiver resultados, consulte as instruções da BIOS ou da UEFI sobre como habilitar a virtualização para o seu processador.

Os resultados variam dependendo da sua CPU. Eis um exemplo de saída de um computador com uma CPU Intel com extensões de virtualização habilitadas no firmware UEFI:

```
$ grep --color -E "vmx|svm" /proc/cpuinfo
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp
lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
cpuid aperfmpf perf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg
fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer
aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb
invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid
ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed
adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln
pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
```

- Se o seu processador suporta virtualizações, procure a documentação da sua distribuição para executar um hipervisor KVM.

- Instale os pacotes necessários para executar um hipervisor KVM.

O método vai variar de acordo com a distribuição, mas aqui estão alguns pontos de partida:

Ubuntu – <https://help.ubuntu.com/lts/serverguide/libvirt.html>

Fedora – <https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-virtualization/>

Arch Linux – <https://wiki.archlinux.org/index.php/KVM>

- Se estiver usando um ambiente de desktop gráfico, é recomendado instalar também o aplicativo `virt-manager`, um front-end gráfico que pode ser usado em uma instalação KVM. Isso ajudará na instalação e gerenciamento da máquina virtual.

Mais uma vez, o método vai depender da distribuição. Eis um exemplo usando o Ubuntu:

```
$ sudo apt install virt-manager
```

- Baixe a imagem ISO de uma distribuição Linux à sua escolha e, seguindo a documentação da distribuição, crie uma nova máquina virtual usando essa ISO.

O pacote `virt-manager` facilita muito essa tarefa. No entanto, uma máquina virtual pode ser criada a partir da linha de comando usando o comando `virt-install`. Experimente os dois métodos para entender como as máquinas virtuais são implementadas.



Tópico 103: Comandos GNU e Unix



103.1 Trabalhar na linha de comando

Referência ao LPI objectivo

LPIC-1 version 5.0, Exam 101, Objective 103.1

Peso

4

Áreas chave de conhecimento

- Usar comandos simples de shell e sequências de comandos de apenas uma linha para executar tarefas básicas na linha de comando.
- Usar e modificar o ambiente de shell incluindo definir, fazer referência e exportar variáveis de ambiente.
- Usar e editar o histórico de comandos.
- Invocar comandos de dentro e de fora do caminho definido.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- bash
- echo
- env
- export
- pwd
- set
- unset
- type

- `which`
- `man`
- `uname`
- `history`
- `.bash_history`
- Quoting



103.1 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.1 Trabalho na linha de comando
Lição:	1 de 2

Introdução

É comum que os recém-chegados ao mundo da administração do Linux e do shell Bash se sintam um pouco perdidos longe do conforto de uma interface gráfica. Eles estão habituados a acessar, com o botão direito do mouse, as dicas visuais e informações contextuais disponibilizadas pelos utilitários gráficos de gerenciamento de arquivos. Portanto, é importante aprender rapidamente a dominar o conjunto relativamente pequeno de ferramentas de linha de comando que permitem acessar instantaneamente todos os dados oferecidos por sua antiga interface gráfica - e muito mais.

Obtendo informações sobre o sistema

De olhos arregalados diante do tracinho piscando de um prompt de linha de comando, você provavelmente se pergunta “Onde estou?” Ou, mais precisamente, “Onde estou agora no sistema de arquivos do Linux? E, se eu criar um novo arquivo, onde ele vai parar?” O que você está procurando é o *diretório de trabalho atual*, e o comando `pwd` responderá às suas dúvidas:

```
$ pwd
/home/frank
```

Vamos supor que Frank esteja atualmente logado no sistema e em seu diretório pessoal: `/home/frank/`. Se Frank criar um arquivo vazio usando o comando `touch` sem especificar qualquer outro local no sistema de arquivos, o arquivo será criado em `/home/frank/`. Se usarmos `ls` para listar o conteúdo do diretório, veremos esse novo arquivo:

```
$ touch newfile
$ ls
newfile
```

Além de sua localização no sistema de arquivos, você também pode precisar de informações sobre o sistema Linux que está executando, como por exemplo o número exato da versão de sua distribuição ou a versão do kernel do Linux atualmente carregada. A ferramenta `uname` é a resposta. E, em particular, `uname` mais a opção `-a` ("all").

```
$ uname -a
Linux base 4.18.0-18-generic #19~18.04.1-Ubuntu SMP Fri Apr 5 10:22:13 UTC 2019
x86_64 x86_64 x86_64 GNU/Linux
```

Neste caso, `uname` mostra que a máquina de Frank tem o kernel do Linux versão 4.18.0 instalado e está executando o Ubuntu 18.04 em uma CPU de 64 bits (`x86_64`).

Obtendo informações sobre comandos

Freqüentemente, você encontrará documentações falando sobre comandos do Linux com os quais ainda não está familiarizado. A própria linha de comando oferece todo tipo de informações úteis sobre o que os comandos fazem e como usá-los com eficácia. As referências mais úteis provavelmente estarão nos muitos arquivos do sistema `man`.

Como regra, os desenvolvedores Linux escrevem arquivos `man` e os distribuem junto com os utilitários que criam. Os arquivos `man` são documentos altamente estruturados cujo conteúdo é dividido intuitivamente em cabeçalhos padronizados. Basta digitar `man` seguido do nome de um comando para exibir informações como o nome do comando, uma breve sinopse de seu uso, uma descrição mais detalhada e alguns dados importantes sobre o histórico e as licenças de uso. Eis um exemplo:

```
$ man uname
UNAME(1)           User Commands          UNAME(1)
NAME
      uname - print system information
SYNOPSIS
```

```
uname [OPTION]...
```

DESCRIPTION

Print certain system information. With no OPTION, same as **-s**.

- a, --all**
print all information, in the following order, except omit **-p** and **-i** if unknown:
- s, --kernel-name**
print the kernel name
- n, --nodename**
print the network node hostname
- r, --kernel-release**
print the kernel release
- v, --kernel-version**
print the kernel version
- m, --machine**
print the machine hardware name
- p, --processor**
print the processor type (non-portable)
- i, --hardware-platform**
print the hardware platform (non-portable)
- o, --operating-system**
print the operating system
- help** display this help and exit
- version**
output version information and exit

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<http://www.gnu.org/software/coreutils/>>
 Report uname translation bugs to
<<http://translationproject.org/team/>>

COPYRIGHT

Copyright©2017 Free Software Foundation, Inc. License GPLv3+: GNU
 GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>.
 This is free software: you are free to change and redistribute it.
 There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

arch(1), uname(2)
 Full documentation at: <<http://www.gnu.org/software/coreutils/uname>>
 or available locally via: info '(coreutils) uname invocation'
GNU coreutils 8.28 January 2018 UNAME(1)

O comando `man` só funciona quando fornecemos um nome de comando exato. Porém, se não tiver certeza do nome do comando que deseja pesquisar, use o comando `apropos` para explorar os nomes e

descrições das páginas `man`. Se, por exemplo, você não consegue se lembrar de que `uname` informa a versão atual do kernel do Linux, pode passar a palavra `kernel` para `apropos`. Aparecerão muitas linhas de saída, mas dentre elas haverá o seguinte:

```
$ apropos kernel
systemd-udevd-kernel.socket (8) - Device event managing daemon
uname (2)                      - get name and information about current kernel
urandom (4)                     - kernel random number source devices
```

Caso não precise da documentação completa de um comando, pode obter seus dados básicos rapidamente usando `type`. Neste exemplo, usamos `type` para consultar quatro comandos separados ao mesmo tempo. Os resultados mostram que `cp` (“copy”) é um programa que vive em `/bin/cp` e que `kill` (muda o estado de um processo em execução) é um comando interno do shell (*shell builtin*) – o que significa que é, na verdade, parte do próprio shell Bash:

```
$ type uname cp kill which
uname is hashed (/bin/uname)
cp is /bin/cp
kill is a shell builtin
which is /usr/bin/which
```

Note que, além de ser um comando binário regular como `cp`, `uname` também aparece como “hashed”. A razão para isso é que Frank recentemente usou `uname` e, para aumentar a eficiência do sistema, o comando foi adicionado a uma tabela de hash para ficar mais acessível na próxima vez que for executado. Se Frank rodasse `type uname` após a inicialização do sistema, ele constataria que `type` voltaria a descrever `uname` como um binário regular.

NOTE

Uma maneira mais rápida de limpar a tabela de hash é executar o comando `hash -d`.

Às vezes – principalmente ao trabalhar com scripts automatizados – precisamos de uma fonte mais simples de informações sobre um comando. O comando `which`, que o comando `type` do exemplo anterior rastreou para nós, retorna somente a localização absoluta de um comando. Este exemplo localiza os comandos `uname` e `which`.

```
$ which uname which
/bin/uname
/usr/bin/which
```

NOTE

Se quiser exibir informações sobre comandos internos do shell (“builtin”), use o comando `help`.

Usando o histórico de comandos

Pode acontecer de você pesquisar cuidadosamente o uso adequado de um comando e executá-lo com êxito junto com uma seleção complicada de opções e argumentos. Mas o que ocorre algumas semanas depois, quando você precisa executar o mesmo comando com as mesmas opções e argumentos, mas não consegue se lembrar dos detalhes? Ao invés de recomeçar a pesquisa do zero, vale a pena tentar recuperar o comando original usando `history`.

Digite `history` para exibir os comandos mais recentes em ordem de execução. É fácil pesquisar nesses comandos usando um pipe para canalizar uma string específica para o comando `grep`. O exemplo abaixo procura por qualquer comando que inclua o texto `bash_history`:

```
$ history | grep bash_history
1605 sudo find /home -name ".bash_history" | xargs grep sudo
```

Aqui, um único comando é retornado junto com seu número na seqüência, 1605.

E por falar em `bash_history`, esse é na verdade o nome de um arquivo oculto que costuma estar no diretório inicial do usuário. Por se tratar de um arquivo oculto (indicado pelo ponto que precede seu nome de arquivo), ele só será visível ao se listar o conteúdo do diretório, usando `ls` com o argumento `-a`:

```
$ ls /home/frank
newfile
$ ls -a /home/frank
.  ..  .bash_history  .bash_logout  .bashrc  .profile  .ssh  newfile
```

O que contém o arquivo `.bash_history`? Dê uma olhada: você encontrará ali centenas e centenas de seus comandos recentes. No entanto, você pode se surpreender ao descobrir que alguns de seus comandos *mais* recentes estão ausentes. Isso porque, embora eles sejam instantaneamente adicionados ao banco de dados dinâmico `history`, as últimas adições ao seu histórico de comandos não são gravadas no arquivo `.bash_history` até o encerramento da sessão.

Para aproveitar o conteúdo de `history` e tornar sua experiência na linha de comando muito mais rápida e eficiente, use as setinhas para cima e para baixo de seu teclado. Pressione a tecla para cima várias vezes para preencher a linha de comando com os comandos recentes. Quando chegar ao que está procurando, basta dar Enter para executá-lo. Assim, é fácil recuperar e, se desejado, modificar

comandos diversas vezes durante uma sessão no shell.

Exercícios Guiados

1. Consultando o sistema `man`, descubra como fazer o `apropos` mostrar apenas uma explicação curta de seu uso e em seguida voltar ao shell.

2. Use o sistema `man` para determinar qual licença de copyright é atribuída ao comando `grep`.

Exercícios Exploratórios

- Identifique a arquitetura de hardware e a versão do kernel do Linux que estão sendo usadas no seu computador em um formato de saída fácil de ler.

- Imprima na tela as últimas vinte linhas do banco de dados dinâmico `history` e do arquivo `.bash_history` para compará-los.

- Use a ferramenta `apropos` para identificar a página `man` na qual se encontra o comando necessário para mostrar o tamanho de um dispositivo de bloco físico conectado em bytes, ao invés de megabytes ou gigabytes.

Resumo

Nesta lição, você aprendeu:

- Como obter informações sobre a localização do sistema de arquivos e a pilha de software do sistema operacional.
- Como encontrar ajuda para o uso de comandos.
- Como identificar a localização do sistema de arquivos e os tipos de binários de comandos.
- Como encontrar e reutilizar comandos executados anteriormente.

Os seguintes comandos foram abordados nesta lição:

pwd

Exibe o caminho para o diretório de trabalho atual.

uname

Exibe a arquitetura de hardware do sistema, a versão do kernel do Linux, a distribuição e a versão da distribuição.

man

Acessa os arquivos de ajuda com a documentação do uso dos comandos.

type

Exibe a localização de um ou mais comandos no sistema de arquivos e seu tipo.

which

Exibe a localização de um comando no sistema de arquivos.

history

Exibe ou reutiliza comandos executados anteriormente.

Respostas aos Exercícios Guiados

1. Consultando o sistema `man`, descubra como fazer o `apropos` mostrar apenas uma explicação curta de seu uso e em seguida voltar ao shell.

Rode `man apropos` e desça a sessão “Options” até encontrar o parágrafo `--usage`.

2. Use o sistema `man` para determinar qual licença de copyright é atribuída ao comando `grep`.

Rode `man grep` e desça até a seção “Copyright” do documento. Note que o programa usa um copyright da Free Software Foundation.

Respostas aos Exercícios Exploratórios

- Identifique a arquitetura de hardware e a versão do kernel do Linux que estão sendo usadas no seu computador em um formato de saída fácil de ler.

Rode `man uname`, leia a seção “Description” e identifique os argumentos de comandos que permitem exibir somente os resultados exatos que se deseja. Note que `-v` mostra a versão do kernel e `-i` a plataforma de hardware.

```
$ man uname  
$ uname -v  
$ uname -i
```

- Imprima na tela as últimas vinte linhas do banco de dados dinâmico `history` e do arquivo `.bash_history` para compará-los.

```
$ history 20  
$ tail -n 20 .bash_history
```

- Use a ferramenta `apropos` para identificar a página `man` na qual se encontra o comando necessário para mostrar o tamanho de um dispositivo de bloco físico conectado em bytes, ao invés de megabytes ou gigabytes.

Uma maneira seria executar `apropos` com o string `block`, ler os resultados, notar que `lsblk` lista os dispositivos de bloco (e assim seria a ferramenta mais provável para dar a resposta que buscamos), rodar `man lsblk`, descer pela seção “Description” e notar que `-b` exibe o tamanho de um dispositivo em bytes. Finalmente, executamos `lsblk -b` para ver o que aparece.

```
$ apropos block  
$ man lsblk  
$ lsblk -b
```



103.1 Lição 2

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.1 Trabalho na linha de comando
Lição:	2 de 2

Introdução

Um ambiente de sistema operacional inclui as ferramentas básicas – como shells de linha de comando e, às vezes, uma interface gráfica – necessárias para trabalhar. Mas seu ambiente também virá com um catálogo de atalhos e valores predefinidos. Nesta lição aprenderemos como listar, invocar e gerenciar esses valores.

Encontrando suas variáveis de ambiente

Então, como identificamos os valores atuais para cada uma de nossas variáveis de ambiente? Uma maneira de fazer isso é por meio do comando `env`:

```
$ env
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gdm/Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
```

[...]

Aparecerão muitos resultados—bem mais do que os incluídos no trecho acima. Mas, por enquanto, observe a entrada PATH, que contém os diretórios nos quais seu shell (e outros aplicativos) procura por outros programas sem a necessidade de especificar um caminho completo. Neste caso, seria possível executar um programa binário que reside, digamos, em /usr/local/bin de dentro do seu diretório pessoal, e ele seria executado como se o arquivo fosse local.

Vamos mudar de assunto um pouquinho. O comando echo exibe na tela o que você mandar. Acredite ou não, haverá diversas ocasiões em que será muito útil fazer echo literalmente repetir algo.

```
$ echo "Hi. How are you?"  
Hi. How are you?
```

Mas echo tem outras cartas na manga. Quando você o alimenta com o nome de uma variável de ambiente—e informa que se trata de uma variável, com o prefixo \$—ao invés de apenas exibir o nome da variável, o shell irá expandi-lo, informando o valor. Não tem certeza se o seu diretório favorito está atualmente em PATH? Você pode verificar rapidamente com o echo:

```
$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/  
games:/snap/bin
```

Criando novas variáveis de ambiente

Você pode adicionar suas próprias variáveis personalizadas ao seu ambiente. A maneira mais simples é usar o caractere =. A string à esquerda será o nome da nova variável e a string à direita, seu valor. Depois, alimente echo com o nome da variável para confirmar se funcionou:

```
$ myvar=hello  
$ echo $myvar  
hello
```

NOTE

Note que não há espaços em torno do sinal de igual durante a atribuição de variáveis.

Mas será que funcionou mesmo? Digite bash no terminal para abrir um novo shell. Ele terá exatamente a mesma aparência daquele que acabamos de usar, mas na verdade é um *filho* do original (que chamamos de *pai*). Agora, dentro desse novo shell filho, tente fazer com que o echo opere sua

mágica da mesma maneira de antes. Nada. O que aconteceu?

```
$ bash
$ echo $myvar
$
```

Uma variável criada da maneira como acabamos de fazer só estará disponível localmente – dentro daquela sessão de shell imediata. Se você iniciar um novo shell – ou fechar a sessão usando `exit` – a variável não irá junto. Aqui, se você digitar `exit`, será levado de volta ao shell pai original que, no momento, é onde queremos estar. Execute `echo $myvar` mais uma vez se quiser apenas confirmar que a variável ainda é válida. Depois digite `export myvar` para passar a variável para quaisquer shells filhos que possam ser abertos posteriormente. Experimente: digite `bash` para abrir um novo shell e em seguida use `echo`:

```
$ exit
$ export myvar
$ bash
$ echo $myvar
hello
```

Tudo isso pode parecer bobagem quando estamos criando shells sem um propósito real. Mas é importantíssimo entender como as variáveis do shell são propagadas pelo sistema para quando você começar a escrever scripts sérios.

Removendo as variáveis de ambiente

Quer saber como limpar todas as variáveis efêmeras que você criou? Uma maneira de fazer isso é simplesmente fechar o shell pai – ou reiniciar o computador. Mas existem jeitos mais simples. Como, por exemplo, `unset`. Basta digitar `unset` (sem o `$`) para matar a variável. Comprove com `echo`.

```
$ unset myvar
$ echo $myvar
$
```

Se um comando `unset` existe, pode apostar que também há um comando `set` complementar. A execução de `set` por si só exibirá um monte de saídas, mas nada muito diferente de `env`. Observe a primeira linha da saída obtida ao filtrarmos por `PATH`:

```
$ set | grep PATH
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games:/snap/bin
[...]
```

Qual a diferença entre `set` e `env`? Para nossos propósitos, a principal delas é que `set` exibe todas as variáveis e funções. Para ilustrar isso, vamos criar uma nova variável chamada `mynewvar` e em seguida confirmar se ela está lá:

```
$ mynewvar=goodbye
$ echo $mynewvar
goodbye
```

Se rodarmos `env` usando `grep` para filtrar pela string `mynewvar`, nenhuma saída será exibida. Mas se rodarmos `set` da mesma maneira, veremos nossa variável local.

```
$ env | grep mynewvar
$ set | grep mynewvar
mynewvar=goodbye
```

Usando aspas para escapar dos caracteres especiais

Este é um bom momento para apresentar o problema dos caracteres especiais. Os caracteres alfanuméricos (a-z e 0-9) normalmente são lidos literalmente pelo Bash. Para criar um novo arquivo chamado `myfile`, bastaria digitar `touch myfile` e o Bash saberia o que fazer. Mas se quisermos incluir um caractere especial no nome do arquivo, teríamos um pouco mais de trabalho.

Para ilustrar esse fato, digitaremos `touch` e em seguida o título: `my big file`. O problema é que existem dois espaços entre as palavras que o Bash interpretará. Embora, tecnicamente, espaços não sejam “caracteres”, é assim que eles se comportam, já que o Bash não os lerá literalmente. Se você listar o conteúdo do diretório atual, em vez de um arquivo chamado `my big file`, verá três arquivos chamados, respectivamente, `my`, `big`, e `file`. Isso porque o Bash pensou que você queria criar vários arquivos cujos nomes estavam sendo dados em uma lista:

```
$ touch my big file
$ ls
my big file
```

Os espaços serão interpretados da mesma forma se você excluir (`rm`) os três arquivos com um só comando:

```
$ rm my big file
```

Agora, vamos fazer do jeito certo. Digite `touch` e as três partes do seu nome de arquivo, mas desta vez coloque o nome entre aspas. Desta vez, funciona. Ao listar o conteúdo do diretório, veremos um único arquivo com o nome correto.

```
$ touch "my big file"  
$ ls  
'my big file'
```

Existem outras maneiras de obter o mesmo efeito. As aspas simples, por exemplo, funcionam da mesma forma que as aspas duplas (note que as aspas simples preservam o valor literal de todos os caracteres, ao passo que as aspas duplas preservam todos os caracteres *exceto* \$, `, \ e, em certos casos, !.)

```
$ rm 'my big file'
```

Podemos incluir uma barra invertida antes de cada caractere especial para “escapar” dessa característica e fazer com que o Bash o leia literalmente.

```
$ touch my\ big\ file
```

Exercícios Guiados

1. Use o comando `export` para adicionar um novo diretório ao seu caminho (PATH)—ele não sobreviverá a uma reinicialização.

2. Use o comando `unset` para remover a variável PATH. Tente executar um comando (como `sudo cat /etc/shadow`) usando `sudo`. O que aconteceu? Por quê? (saia do shell para retornar ao estado original.)

Exercícios Exploratórios

1. Procure na internet a lista completa de caracteres especiais e explore-os.
2. Tente executar comandos usando strings compostas de caracteres especiais e diversos métodos para escapar deles. Existem diferenças de comportamento entre esses métodos?

Resumo

Nesta lição, você aprendeu:

- Como identificar as variáveis de ambiente de seu sistema.
- Como criar suas próprias variáveis de ambiente e exportá-la para outros shells.
- Como remover variáveis de ambiente e usar os comandos `env` e `set`.
- Como escapar dos caracteres especiais para que o Bash os leia literalmente.

Os seguintes comandos foram abordados nesta lição:

echo

Exibe as strings e variáveis informadas.

env

Entende e modifica suas variáveis de ambiente.

export

Passa uma variável de ambiente para os shells filhos.

unset

Desconfigura os valores e atributos das variáveis e funções do shell.

Respostas aos Exercícios Guiados

1. Use o comando `export` para adicionar um novo diretório ao seu caminho (PATH)—ele não sobreviverá a uma reinicialização.

Podemos adicionar temporariamente um novo diretório (por exemplo, um chamado `myfiles` dentro do diretório inicial) ao caminho usando `export PATH="/home/yourname/myfiles:$PATH"`. Crie um script simples no diretório `myfiles/`, torne-o executável e tente executá-lo a partir de um diretório diferente. Estes comandos pressupõem que você esteja no diretório inicial, que contém um diretório chamado `myfiles`.

```
$ touch myfiles/myscript.sh
$ echo '#!/bin/bash' >> myfiles/myscript.sh
$ echo 'echo Hello' >> myfiles/myscript.sh
$ chmod +x myfiles/myscript.sh
$ myscript.sh
Hello
```

2. Use o comando `unset` para remover a variável PATH. Tente executar um comando (como `sudo cat /etc/shadow`) usando `sudo`. O que aconteceu? Por quê? (saia do shell para retornar ao estado original.)

O comando `unset PATH` apaga as configurações de caminho atuais. Não será possível invocar um binário sem seu endereço absoluto. Por essa razão, se tentarmos rodar um comando usando `sudo` (que por sua vez é um programa binário localizado em `/usr/bin/sudo`) a operação falhará—a menos que especifiquemos a localização absoluta, como em: `/usr/bin/sudo /bin/cat /etc/shadow`. Podemos redefinir o PATH usando `export` ou simplesmente saindo do shell.

Respostas aos Exercícios Exploratórios

1. Procure na internet a lista completa de caracteres especiais e explore-os.

Eis uma lista: & ; | * ? " ' [] () \$ < > { } # / \ ! ~.

2. Tente executar comandos usando strings compostas de caracteres especiais e diversos métodos para escapar deles. Existem diferenças de comportamento entre esses métodos?

O escape com `"` preserva os valores especiais do cifrão, da crase e da barra invertida. Já o escape com o `'` faz com que *todos* os caracteres sejam interpretados literalmente.

```
$ echo "$mynewvar"  
goodbye  
$ echo '$mynewvar'  
$mynewvar
```



103.2 Processar fluxos de texto usando filtros

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.2

Peso

2

Áreas chave de conhecimento

- Enviar arquivos de texto e saídas de fluxo de textos através de filtros para modificar a saída usando comandos padrão UNIX encontrados no pacote GNU textutils.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- bzcat
- cat
- cut
- head
- less
- md5sum
- nl
- od
- paste
- sed
- sha256sum
- sha512sum

- sort
- split
- tail
- tr
- uniq
- wc
- xzcat
- zcat



103.2 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.2 Processar fluxos de texto usando filtros
Lição:	1 de 1

Introdução

Lidar com textos é uma parte importante do trabalho de todo administrador de sistemas. Doug McIlroy, membro da equipe de desenvolvimento original do Unix, resumiu a filosofia Unix e disse (entre outras coisas importantes): “Escreva programas para lidar com fluxos de texto, porque essa é uma interface universal.” O Linux é inspirado no sistema operacional Unix e defende essa mesma filosofia; portanto, um administrador deve estar pronto para lidar com muitas ferramentas de manipulação de texto dentro de uma distribuição Linux.

Uma revisão rápida sobre redirecionamentos e pipes

Também da filosofia Unix:

- Escreva programas que façam apenas uma coisa e a façam bem.
- Escreva programas que trabalhem juntos.

Uma das principais formas de fazer os programas trabalharem juntos é usar *piping* (canalização) e *redirecionamentos*. Praticamente todos os programas de manipulação de texto obtêm texto de uma entrada padrão (*stdin*), produzem uma saída padrão (*stdout*) e enviam eventuais erros para uma saída

de erro padrão (*stderr*). A menos que especifiquemos o contrário, a entrada padrão será o que digitamos no teclado (o programa lê esse texto quando pressionamos Enter). Da mesma forma, a saída padrão e os erros serão exibidos na tela do terminal. Vamos ver como isso funciona.

Em seu terminal, digite `cat` e pressione a tecla Enter. Em seguida, digite algum texto aleatório.

```
$ cat
This is a test
This is a test
Hey!
Hey!
It is repeating everything I type!
It is repeating everything I type!
(I will hit ctrl+c so I will stop this nonsense)
(I will hit ctrl+c so I will stop this nonsense)
^C
```

Para saber mais sobre o comando `cat` (o termo vem de “concatenar”), consulte as páginas de manual.

NOTE

Se estiver trabalhando em uma instalação simples de um servidor Linux, alguns comandos como `info` e `less` talvez não estejam disponíveis. Se for este o caso, instale essas ferramentas usando o procedimento adequado em seu sistema, conforme descrito nas lições correspondentes.

Conforme demonstrado acima, se você não especificar de onde `cat` deve ler, ele lerá a entrada padrão (o que você digitar) e produzirá o que for lido na janela do terminal (a saída padrão).

Agora tente o seguinte:

```
$ cat > mytextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
^C

$ cat mytextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

O sinal de `>` (maior que) diz ao `cat` para enviar a saída ao arquivo `mytextfile`, não para a saída

padrão. Agora tente o seguinte:

```
$ cat mytextfile > mynewtextfile
$ cat mynewtextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

O efeito desse comando é copiar `mytextfile` para `mynewtextfile`. Na verdade, podemos verificar se esses dois arquivos têm o mesmo conteúdo executando um `diff`:

```
$ diff mynewtextfile mytextfile
```

Como não há saída, os arquivos são idênticos. Agora, tente o operador de redirecionamento de acréscimo (`>>`):

```
$ echo 'This is my new line' >> mynewtextfile
$ diff mynewtextfile mytextfile
4d3
< This is my new line
```

Até agora, usamos redirecionamentos para criar e manipular arquivos. Também podemos usar pipes (representados pelo símbolo `|`) para redirecionar a saída de um programa para outro programa. Vamos encontrar as linhas onde a palavra “this” aparece:

```
$ cat mytextfile | grep this
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this

$ cat mytextfile | grep -i this
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

Nós canalizamos a saída de `cat` para outro comando: `grep`. Observe que, quando ignoramos o uso de maiúsculas e minúsculas (usando a opção `-i`), obtemos uma linha extra como resultado.

Processando fluxos de texto

Lendo um arquivo compactado

Vamos criar um arquivo chamado `ftu.txt` contendo uma lista dos seguintes comandos:

```
bzcat  
cat  
cut  
head  
less  
md5sum  
nl  
od  
paste  
sed  
sha256sum  
sha512sum  
sort  
split  
tail  
tr  
uniq  
wc  
xzcat  
zcat
```

A seguir, vamos usar o comando `grep` para imprimir todas as linhas que contêm a string `cat`:

```
$ cat ftu.txt | grep cat  
bzcat  
cat  
xzcat  
zcat
```

Outra forma de obter essas informações é apenas usar o comando `grep` para filtrar o texto diretamente, sem a necessidade de usar outro aplicativo para enviar o fluxo de texto para `stdout`.

```
$ grep cat ftu.txt  
bzcat  
cat  
xzcat
```

```
zcat
```

NOTE Lembre-se de que há muitas maneiras de realizar uma tarefa usando o Linux.

Existem outros comandos que lidam com arquivos compactados (`bzcat` para arquivos compactados `bzip`, `xzcat` para arquivos `xz` e `zcat` para arquivos `gzip`). Eles são usados para visualizar o conteúdo de um arquivo compactado com base no algoritmo de compactação empregado.

Verifique se o arquivo recém-criado `ftu.txt` é o único no diretório e, em seguida, crie uma versão compactada `gzip` do arquivo:

```
$ ls ftu*
ftu.txt

$ gzip ftu.txt
$ ls ftu*
ftu.txt.gz
```

Em seguida, use o comando `zcat` para visualizar o conteúdo do arquivo compactado com o `gzip`:

```
$ zcat ftu.txt.gz
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzcat
zcat
```

Note que `gzip` comprime `ftu.txt` em `ftu.txt.gz` e remove o arquivo original. Por padrão, nenhuma saída do comando `gzip` é exibida. Porém, se você deseja que o `gzip` lhe diga o que está fazendo, use a opção `-v` para a saída “verbosa”.

Visualizando um arquivo no paginador

Sabemos que `cat` concatena um arquivo para a saída padrão (quando um arquivo é informado após o comando). O arquivo `/var/log/syslog` é onde o sistema Linux armazena tudo de importante que acontece no sistema. Usamos o comando `sudo` para elevar nossos privilégios e poder ler o arquivo `/var/log/syslog`:

```
$ sudo cat /var/log/syslog
```

...e veremos mensagens rolando muito rápido na janela do terminal. É possível canalizar a saída para o programa `less` para que os resultados sejam paginados. Com o `less`, podemos usar as setinhas do teclado para navegar pela saída. Comandos do tipo `vi` também ajudam a navegar e pesquisar em todo o texto.

No entanto, em vez de canalizar o comando `cat` para um programa de paginação, é mais pragmático usar o programa de paginação diretamente:

```
$ sudo less /var/log/syslog  
... (saída omitida para maior clareza)
```

Obtendo uma parte de um arquivo de texto

Se apenas o início ou o final de um arquivo precisar ser analisado, existem outros métodos disponíveis. O comando `head` é usado para ler as primeiras dez linhas de um arquivo por padrão, e o comando `tail` é usado para ler as dez linhas últimas. Experimente:

```
$ sudo head /var/log/syslog  
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0"  
x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed  
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.  
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.  
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,  
tid=928, prio=low)  
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'  
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'  
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
```

```

Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
$ sudo tail /var/log/syslog
Nov 13 10:24:45 hypatia kernel: [ 8001.679238] mce: CPU7: Core temperature/speed
normal
Nov 13 10:24:46 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating
via systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425
comm="/usr/lib/tracker/tracker-miner-fs ")
Nov 13 10:24:46 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:24:47 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:24:47 hypatia systemd[2004]: Started Tracker metadata extractor.
Nov 13 10:24:54 hypatia kernel: [ 8010.462227] mce: CPU0: Core temperature above
threshold, cpu clock throttled (total events = 502907)
Nov 13 10:24:54 hypatia kernel: [ 8010.462228] mce: CPU4: Core temperature above
threshold, cpu clock throttled (total events = 502911)
Nov 13 10:24:54 hypatia kernel: [ 8010.469221] mce: CPU0: Core temperature/speed
normal
Nov 13 10:24:54 hypatia kernel: [ 8010.469222] mce: CPU4: Core temperature/speed
normal
Nov 13 10:25:03 hypatia systemd[2004]: tracker-extract.service: Succeeded.

```

Para ajudar a ilustrar o número de linhas exibidas, podemos canalizar a saída do comando `head` para o comando `nl`, que exibirá o número de linhas de texto transmitidas para o comando:

```

$ sudo head /var/log/syslog | nl
1 Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd"
swVersion="8.1910.0" x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was
HUPed
2 Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
3 Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
4 Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started
(pid=882, tid=928, prio=low)
5 Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
6 Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
7 Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
8 Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
9 Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
10 Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,

```

```
tid=929, prio=high)
```

E podemos fazer o mesmo canalizando a saída do comando `tail` para o comando `wc`, que por padrão conta o número de palavras em um documento, e usando a opção `-l` para imprimir na tela o número de linhas de texto lidas pelo comando:

```
$ sudo tail /var/log/syslog | wc -l
10
```

Caso um administrador precise revisar mais (ou menos) linhas do início ou do fim de um arquivo, a opção `-n` pode ser usada para limitar a saída dos comandos:

```
$ sudo tail -n 5 /var/log/syslog
Nov 13 10:37:24 hypatia systemd[2004]: tracker-extract.service: Succeeded.
Nov 13 10:37:42 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating
via systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425
comm="/usr/lib/tracker/tracker-miner-fs ")
Nov 13 10:37:42 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:37:43 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:37:43 hypatia systemd[2004]: Started Tracker metadata extractor.
$ sudo head -n 12 /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0"
x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,
tid=928, prio=low)
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
Nov 12 08:04:30 hypatia vdr: [882] no DVB device found
Nov 12 08:04:30 hypatia vdr: [882] initializing plugin: vnsiserver (1.8.0): VDR-
Network-Streaming-Interface (VNSI) Server
```

Noções básicas de sed, o editor de fluxo

Vamos dar uma olhada em outros arquivos, termos e utilitários que não têm `cat` em seus nomes. Podemos fazer isso passando a opção `-v` para `grep`, que instrui o comando a exibir apenas as linhas que não contêm `cat`:

```
$ zcat ftu.txt.gz | grep -v cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Muito do que podemos fazer com `grep`, também podemos fazer com `sed`—o editor de fluxo para filtrar e transformar texto (como consta na página de manual do `sed`). Primeiro, vamos recuperar nosso arquivo `ftu.txt` descompactando o pacote `gzip` do arquivo:

```
$ gunzip ftu.txt.gz
$ ls ftu*
ftu.txt
```

Em seguida, podemos usar `sed` para listar apenas as linhas que contêm a string `cat`:

```
$ sed -n /cat/p < ftu.txt
bzcat
cat
xzcat
zcat
```

Usamos o sinal de menor que `<` para direcionar o conteúdo do arquivo `ftu.txt` into para o comando

sed. A palavra entre barras (isto é, /cat/) é o termo que estamos procurando. A opção -n instrui o sed a não produzir nenhuma saída (a não ser as saídas solicitadas pelo comando p). Tente executar este mesmo comando sem a opção -n para ver o que acontece. Depois, tente o seguinte:

```
$ sed /cat/d < ftu.txt
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Sem a opção -n, sed imprime na tela tudo o que está no arquivo, exceto o que d instrui sed a remover da saída.

Um uso comum de sed é localizar e substituir texto em um arquivo. Suponha que você queira alterar todas as ocorrências de cat para dog. Para isso, use o sed, instruindo a opção s a trocar cada instância do primeiro termo, cat, pelo segundo, dog:

```
$ sed s/cat/dog/ < ftu.txt
bzdog
dog
ct
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
```

```
sort
split
tail
tr
uniq
wc
xzdog
zdog
```

Em vez de usar um operador de redirecionamento (<) para passar o arquivo `ftu.txt` para o comando `sed`, podemos fazer o comando `sed` operar diretamente no arquivo. Tentaremos isso a seguir, enquanto criamos simultaneamente um backup do arquivo original:

```
$ sed -i.backup s/cat/dog/ ftu.txt
$ ls ftu*
ftu.txt  ftu.txt.backup
```

A opção `-i` executa uma operação `sed` local no arquivo *original*. Se você não usar `.backup` após o parâmetro `-i`, vai reescrever o arquivo *original*. Qualquer texto inserido após o parâmetro `-i` será o nome com o qual o arquivo original será salvo antes das modificações que você pediu ao `sed` para realizar.

Garantindo a integridade dos dados

Como vimos, é fácil manipular arquivos no Linux. Há momentos em que queremos compartilhar um arquivo com outra pessoa e ter certeza de que o destinatário receberá uma cópia fiel do arquivo original. Um uso muito comum dessa técnica é quando os servidores das distribuições Linux hospedam imagens de CD ou DVD de seu software para download, juntamente com arquivos que contêm os valores calculados da soma de verificação (checksum) dessas imagens de disco. Eis, por exemplo, a listagem de um mirror de download do Debian:

```
...[PARENTDIR] Parent Directory ...
[SUM]      MD5SUMS                      2019-09-08 17:46 274
[CRT]      MD5SUMS.sign                  2019-09-08 17:52 833
[SUM]      SHA1SUMS                     2019-09-08 17:46 306
[CRT]      SHA1SUMS.sign                2019-09-08 17:52 833
[SUM]      SHA256SUMS                   2019-09-08 17:46 402
[CRT]      SHA256SUMS.sign              2019-09-08 17:52 833
[SUM]      SHA512SUMS                   2019-09-08 17:46 658
[CRT]      SHA512SUMS.sign              2019-09-08 17:52 833
```

```
[ISO]      debian-10.1.0-amd64-netinst.iso    2019-09-08 04:37 335M
[ISO]      debian-10.1.0-amd64-xfce-CD-1.iso  2019-09-08 04:38 641M
[ISO]      debian-edu-10.1.0-amd64-netinst.iso 2019-09-08 04:38 405M
[ISO]      debian-mac-10.1.0-amd64-netinst.iso  2019-09-08 04:38 334M
.....
```

Na listagem acima, os arquivos de imagem do instalador do Debian são acompanhados por arquivos de texto que contêm checksums dos arquivos dos vários algoritmos (MD5, SHA1, SHA256 e SHA512).

NOTE

Uma soma de verificação (checksum) é um valor derivado de um cálculo matemático, baseado em uma função hash criptográfica, em relação a um arquivo. Existem diferentes tipos de funções hash criptográficas com diferentes intensidades. Para o exame, você deverá estar familiarizado com o uso de `md5sum`, `sha256sum` e `sha512sum`.

Depois de baixar um arquivo (por exemplo, a imagem `debian-10.1.0-amd64-netinst.iso`), você deve comparar a soma de verificação do arquivo baixado com o valor de soma de verificação que lhe foi fornecido.

Eis um exemplo ilustrativo. Vamos calcular o valor SHA256 do arquivo `ftu.txt` usando o comando `sha256sum`:

```
$ sha256sum ftu.txt
345452304fc26999a715652543c352e5fc7ee0c1b9deac6f57542ec91daf261c ftu.txt
```

A longa sequência de caracteres que precede o nome do arquivo é o valor do checksum SHA256 desse arquivo de texto. Vamos criar um arquivo contendo esse valor, que servirá para verificar a integridade do arquivo de texto original. Usamos para isso o mesmo comando `sha256sum` e redirecionamos a saída para um arquivo:

```
$ sha256sum ftu.txt > sha256.txt
```

Agora, para verificar o arquivo `ftu.txt`, basta usar o mesmo comando e fornecer o nome do arquivo que contém o valor do checksum junto com a opção `-c`:

```
$ sha256sum -c sha256.txt
ftu.txt: OK
```

O valor contido no arquivo corresponde à soma de verificação SHA256 calculada para nosso arquivo `ftu.txt`, exatamente como esperávamos. No entanto, se o arquivo original for modificado (por exemplo, alguns bytes perdidos durante o download, ou uma adulteração deliberada), a verificação do valor falhará. Nesse caso, sabemos que o arquivo está danificado ou corrompido e não podemos confiar na integridade de seu conteúdo. Para demonstrar esse ponto, vamos adicionar algum texto no final do arquivo:

```
$ echo "new entry" >> ftu.txt
```

A seguir, tentaremos verificar a integridade do arquivo:

```
$ sha256sum -c sha256.txt
ftu.txt: FAILED
sha256sum: WARNING: 1 computed checksum did NOT match
```

Vemos assim que a soma de verificação não corresponde ao esperado. Portanto, não podemos confiar na integridade deste arquivo. Poderíamos tentar baixar uma nova cópia, relatar a falha da soma de verificação ao remetente do arquivo ou contatar a equipe de segurança do data center, dependendo da importância do arquivo.

Um olhar mais aprofundado nos arquivos

O comando octal dump (`od`) é freqüentemente usado para depurar aplicativos e diversos tipos de arquivos. Por si só, o comando `od` somente lista o conteúdo de um arquivo em formato octal. Podemos usar nosso arquivo `ftu.txt` para praticar com esse comando:

```
$ od ftu.txt
0000000 075142 060543 005164 060543 005164 072543 005164 062550
0000020 062141 066012 071545 005163 062155 071465 066565 067012
0000040 005154 062157 070012 071541 062564 071412 062145 071412
0000060 060550 032462 071466 066565 071412 060550 030465 071462
0000100 066565 071412 071157 005164 070163 064554 005164 060564
0000120 066151 072012 005162 067165 070551 073412 005143 075170
0000140 060543 005164 061572 072141 000012
0000151
```

A primeira coluna é o *deslocamento de bytes* para cada linha da saída. Como por padrão `od` imprime informações no formato octal, cada linha começa com um deslocamento de bytes de oito bits, seguido por oito colunas, cada uma contendo o valor octal dos dados dentro dessa coluna.

TIP Lembre que um *byte* tem 8 bits.

Se precisar visualizar o conteúdo de um arquivo em formato hexadecimal, use a opção `-x`:

```
$ od -x ftu.txt
0000000 7a62 6163 0a74 6163 0a74 7563 0a74 6568
0000020 6461 6c0a 7365 0a73 646d 7335 6d75 6e0a
0000040 0a6c 646f 700a 7361 6574 730a 6465 730a
0000060 6168 3532 7336 6d75 730a 6168 3135 7332
0000100 6d75 730a 726f 0a74 7073 696c 0a74 6174
0000120 6c69 740a 0a72 6e75 7169 770a 0a63 7a78
0000140 6163 0a74 637a 7461 000a
0000151
```

Nesse caso, cada uma das oito colunas após o deslocamento de bytes é representada por seus equivalentes hexadecimais.

Um uso útil do comando `od` é depurar scripts. Por exemplo, o comando `od` pode nos mostrar caracteres que não são normalmente visíveis e que existem em um arquivo, como indicadores de *nova linha*. Usamos para isso a opção `-c`, de modo que, em vez de exibir a notação numérica para cada byte, as colunas serão mostradas como seus equivalentes em caracteres:

```
$ od -c ftu.txt
0000000 b z c a t \n c a t \n c u t \n h e
0000020 a d \n l e s s \n m d 5 s u m \n n
0000040 l \n o d \n p a s t e \n s e d \n s
0000060 h a 2 5 6 s u m \n s h a 5 1 2 s
0000100 u m \n s o r t \n s p l i t \n t a
0000120 i l \n t r \n u n i q \n w c \n x z
0000140 c a t \n z c a t \n
0000151
```

Todas as ocorrências de nova linha no arquivo são representadas pelos caracteres ocultos `\n`. Se você deseja apenas visualizar todos os caracteres em um arquivo e não precisa ver as informações de deslocamento de bytes, pode remover essa coluna da saída da seguinte forma:

```
$ od -An -c ftu.txt
b z c a t \n c a t \n c u t \n h e
a d \n l e s s \n m d 5 s u m \n n
l \n o d \n p a s t e \n s e d \n s
h a 2 5 6 s u m \n s h a 5 1 2 s
```

```
u m \n s o r t \n s p l i t \n t a
i l \n t r \n u n i q \n w c \n x z
c a t \n z c a t \n
```

Exercícios Guiados

- Alguém acaba de doar um laptop para sua escola e você deseja instalar Linux nele. Ele veio sem manual e você é obrigado a inicializá-lo a partir de um pen drive USB sem nenhuma interface gráfica. Aparece um terminal shell e você sabe que, para cada processador presente, haverá uma linha dedicada no arquivo /proc/cpuinfo:

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model      : 158
```

(linhas puladas)

```
processor : 1
vendor_id : GenuineIntel
cpu family : 6
model      : 158
```

(mais linhas puladas)

- Usando os comandos grep e wc, exiba o número de processadores presentes.

- Faça o mesmo com sed em vez de grep.

- Explore seu arquivo local /etc/passwd com os comandos grep, sed, head e tail de acordo com as tarefas descritas abaixo:

- Quais usuários têm acesso a um shell Bash?

- Muitos dos usuários de seu sistema existem para lidar com programas específicos ou para fins administrativos. Eles não têm acesso a um shell. Quantos deles estão presentes no sistema?

- Quantos usuários e grupos existem em seu sistema (lembre-se: use apenas o arquivo /etc/passwd)?

- Liste apenas a primeira, a última e a décima linha do arquivo /etc/passwd.

3. Considere este arquivo de exemplo /etc/passwd. Copie as linhas abaixo para um arquivo local chamado `mypasswd` para este exercício.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence
Daemon,,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- Liste todos os usuários no grupo 1000 (use sed para selecionar apenas o campo apropriado) do arquivo mypasswd.

- Liste apenas o nome completo de todos os usuários deste grupo (use sed e cut).

Exercícios Exploratórios

1. Usando novamente o arquivo `mypasswd` dos exercícios anteriores, imagine um comando Bash que selecione um indivíduo do Main Office para ganhar uma rifa. Use o comando `sed` para imprimir apenas as linhas do Main Office e, em seguida, uma sequência de comando `cut` para recuperar o primeiro nome de cada usuário a partir dessas linhas. Depois, classifique esses nomes aleatoriamente e imprima apenas o nome principal da lista.

2. Quantas pessoas trabalham em Finance, Engineering e Sales? Pense em explorar o comando `uniq`.

3. Agora, vamos preparar um arquivo CSV (valores separados por vírgula) para poder importar facilmente, do arquivo `mypasswd` do exemplo anterior, o arquivo `names.csv` para o LibreOffice. O conteúdo do arquivo terá o seguinte formato:

```
First Name,Last Name,Position  
Carol,Smith,Finance  
...  
John,Chapel,Sales
```

Dica: use os comandos `sed`, `cut` e `paste` para obter os resultados desejados. Note que a vírgula (,) será o delimitador desse arquivo.

4. Suponha que a planilha `names.csv` criada no exercício anterior seja um arquivo importante e queremos ter certeza de que ninguém vai adulterá-lo desde o momento do envio até a recepção pelo destinatário. Como podemos garantir a integridade desse arquivo usando `md5sum`?

5. Você prometeu a si mesmo que lerá 100 linhas por dia de um livro clássico e decidiu começar com *Mariner and Mystic* de Herman Melville. Desenvolva um comando usando `split` para dividir este livro em seções de 100 linhas cada. Para obter o livro em formato de texto simples, pesquise em <https://www.gutenberg.org>.

6. Usando `ls -l` no diretório `/etc`, que tipo de listagem obtemos? Usando o comando `cut` na saída do comando `ls` fornecido, como exibir apenas os nomes dos arquivos? E quanto ao nome dos arquivos e seu proprietário? Junto com os comandos `ls -l` e `cut`, utilize o comando `tr` para

espremer diversas ocorrências de um espaço em um único espaço para auxiliar na formatação da saída com um comando `cut`.

7. Este exercício pressupõe que você está em uma máquina real (não virtual). Também é preciso ter um pendrive à mão. Releia as páginas de manual do comando `tail` e descubra como seguir um arquivo conforme adicionamos texto a ele. Enquanto monitora a saída de um comando `tail` no arquivo `/var/log/syslog`, insira um pendrive. Escreva o comando completo que usaria para obter o Produto (Product), o Fabricante (Manufacturer) e a quantidade total de memória (Blocks) do seu pendrive.
-

Resumo

Lidar com fluxos de texto é de grande importância ao administrar qualquer sistema Linux. Os fluxos de texto podem ser processados usando scripts para automatizar tarefas diárias ou localizar informações de depuração relevantes em arquivos de log. Eis um breve resumo dos comandos abordados nesta lição:

cat

Usado para combinar ou ler arquivos de texto simples.

bzcat

Permite processar ou ler arquivos comprimidos usando o método `bzip2`.

xzcat

Permite processar ou ler arquivos comprimidos usando o método `xz`.

zcat

Permite processar ou ler arquivos comprimidos usando o método `gzip`.

less

Este comando pagina o conteúdo de um arquivo e possibilita a funcionalidade de navegação e pesquisa.

head

Este comando exibe as primeiras 10 linhas de um arquivo por padrão. A opção `-n` permite exibir mais ou menos linhas.

tail

Este comando exibe as últimas 10 linhas de um arquivo por padrão. A opção `-n` permite exibir mais ou menos linhas. A opção `-f` é usada para seguir a saída de um arquivo de texto conforme novos dados são escritos nele.

wc

Abreviação de “word count” (contagem de palavras), mas dependendo dos parâmetros usados ele conta caracteres, palavras e linhas.

sort

Usado para organizar a saída de uma listagem em ordem alfabética, alfabética inversa ou aleatória.

uniq

Usado para listar (e contar) strings correspondentes.

od

O comando “octal dump” é usado para exibir um arquivo binário em notação octal, decimal ou hexadecimal.

nl

O comando “number line” exibe o número de linhas em um arquivo, além de recriar um arquivo com cada linha prefixada por seu número de linha.

sed

O editor de fluxo pode ser usado para encontrar ocorrências correspondentes de strings usando Expressões Regulares, bem como editar arquivos usando padrões predefinidos.

tr

O comando translate substitui caracteres e também remove e compacta caracteres repetidos.

cut

Este comando imprime colunas de arquivos de texto como campos baseados no delimitador de caracteres de um arquivo.

paste

Une arquivos em colunas com base no uso de separadores de campo.

split

Este comando divide arquivos maiores em menores conforme os critérios definidos nas opções do comando.

md5sum

Usado para calcular o valor de hash MD5 de um arquivo. Também usado para verificar um arquivo em relação a um valor de hash existente para garantir a integridade do arquivo.

sha256sum

Usado para calcular o valor de hash SHA256 de um arquivo. Também usado para verificar um arquivo em relação a um valor de hash existente para garantir a integridade do arquivo.

sha512sum

Usado para calcular o valor de hash SHA512 de um arquivo. Também usado para verificar um

arquivo em relação a um valor de hash existente para garantir a integridade do arquivo.

Respostas aos Exercícios Guiados

- Alguém acaba de doar um laptop para sua escola e você deseja instalar Linux nele. Ele veio sem manual e você é obrigado a inicializá-lo a partir de um pen drive USB sem nenhuma interface gráfica. Aparece um terminal shell e você sabe que, para cada processador presente, haverá uma linha dedicada no arquivo /proc/cpuinfo:

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model      : 158
```

(linhas puladas)

```
processor : 1
vendor_id : GenuineIntel
cpu family : 6
model      : 158
```

(more lines skipped)

- Usando os comandos grep e wc, exiba o número de processadores presentes.

Eis duas opções:

```
$ cat /proc/cpuinfo | grep processor | wc -l
$ grep processor /proc/cpuinfo | wc -l
```

Agora que você sabe que existem várias maneiras de fazer a mesma coisa, quando deve usar uma ou outra? Isso depende de vários fatores, sendo os dois mais importantes o desempenho e a legibilidade. Na maioria das vezes, usamos comandos de shell dentro de scripts de shell para automatizar tarefas; quanto maiores e mais complexos seus scripts se tornam, mais precisamos nos preocupar em mantê-los rápidos.

- Faça o mesmo com sed em vez de grep.

Agora, em vez de grep vamos tentar isso com sed:

```
$ sed -n '/processor/p' /proc/cpuinfo | wc -l
```

Aqui, usamos `sed` com o parâmetro `-n` para que `sed` não imprima nada, exceto o que corresponde à expressão `processor`, conforme instruído pelo comando `p`. Como fizemos nas soluções de `grep`, `wc -l` conta o número de linhas e, portanto, o número de processadores presentes.

Estude o exemplo a seguir:

```
$ sed -n /processor/p /proc/cpuinfo | sed -n '$='
```

Esta sequência de comando fornece resultados idênticos aos do exemplo anterior, no qual a saída de `sed` foi canalizada para um comando `wc`. A diferença, aqui, é que em vez de usar `wc -l` para contar o número de linhas, `sed` é novamente invocado para fornecer uma funcionalidade equivalente. Mais uma vez, estamos suprimindo a saída de `sed` com a opção `-n`, exceto para a expressão que estamos chamando explicitamente, que é '`=$`'. Essa expressão diz ao `sed` para encontrar uma correspondência para a última linha (`$`) e então imprimir o número dessa linha (`=`).

2. Explore seu arquivo local `/etc/passwd` com os comandos `grep`, `sed`, `head` e `tail` de acordo com as tarefas descritas abaixo:

- Quais usuários têm acesso a um shell Bash?

```
$ grep ":/bin/bash$" /etc/passwd
```

Vamos refinar esta resposta exibindo apenas o nome do usuário que utiliza o shell Bash.

```
$ grep ":/bin/bash$" /etc/passwd | cut -d: -f1
```

O nome do usuário é o primeiro campo (parâmetro `-f1` do comando `cut`) e o arquivo `/etc/passwd` usa `:` como separadores (`-d:` parâmetro do comando `cut`). Nós apenas canalizamos a saída do comando `grep` para o comando `cut` apropriado.

- Muitos dos usuários de seu sistema existem para lidar com programas específicos ou para fins administrativos. Eles não têm acesso a um shell. Quantos deles estão presentes no sistema?

A maneira mais fácil de descobrir isso é exibir as linhas correspondentes às contas que não usam o shell Bash:

```
$ grep -v ":/bin/bash$" /etc/passwd | wc -l
```

- Quantos usuários e grupos existem em seu sistema (lembre-se: use apenas o arquivo /etc/passwd)?

O primeiro campo de qualquer linha do arquivo /etc/passwd é o nome do usuário, o segundo é tipicamente um x indicando que a senha do usuário não está armazenada aqui (ela é criptografada no arquivo /etc/shadow). O terceiro é o id do usuário (UID) e o quarto é o id do grupo (GID). Portanto, isso deve nos fornecer o número de usuários:

```
$ cut -d: -f3 /etc/passwd | wc -l
```

Bem, isso funciona na maioria dos casos. No entanto, há situações em que definimos diferentes superusuários ou outros tipos especiais de usuários que compartilham o mesmo UID (id de usuário). Assim, para ter certeza, vamos canalizar o resultado do comando `cut` para o comando `sort` e então contar o número de linhas.

```
$ cut -d: -f3 /etc/passwd | sort -u | wc -l
```

Agora, para o número de grupos:

```
$ cut -d: -f4 /etc/passwd | sort -u | wc -l
```

- Liste apenas a primeira, a última e a décima linha do arquivo /etc/passwd.

Podemos fazer assim:

```
$ sed -n -e '1'p -e '10'p -e '$'p /etc/passwd
```

Lembre-se de que o parâmetro `-n` diz ao `sed` para não imprimir nada além do que é especificado pelo comando `p`. O cifrão (\$) usado aqui é uma expressão regular que representa a última linha do arquivo.

- Considere este exemplo do arquivo /etc/passwd. Copie as linhas abaixo para um arquivo local chamado mypasswd para este exercício.

```

root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence
Daemon,,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash

```

- Liste todos os usuários no grupo 1000 (use sed para selecionar apenas o campo apropriado) do arquivo mypasswd.

O GID é o quarto campo do arquivo /etc/passwd. Você pode ter vontade de tentar o seguinte:

```
$ sed -n /1000/p mypasswd
```

Nesse caso, você obterá também esta linha:

```
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
```

Sabemos que não está correto, pois Carol Smith é membro do GID 2000 e a correspondência foi encontrada por causa do UID. No entanto, você deve ter notado que, após o GID, o campo seguinte começa com um caractere maiúsculo. Podemos usar uma expressão regular para resolver esse problema.

```
$ sed -n /:1000:[A-Z]/p mypasswd
```

A expressão [A-Z] busca por quaisquer caracteres em maiúsculas. Vamos falar mais sobre isso na lição correspondente.

- Liste apenas o nome completo de todos os usuários deste grupo (use `sed` e `cut`).

Use a mesma técnica empregada para resolver a primeira parte deste exercício e canalize para um comando `cut`.

```
$ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5  
Dave Edwards,Finance,,,Main Office  
Emma Jones,Finance,,,Main Office  
Frank Cassidy,Finance,,,Main Office  
Grace Kearns,Engineering,,,Main Office  
Henry Adams,Sales,,,Main Office  
John Chapel,Sales,,,Main Office
```

Ainda não chegamos lá! Note como os campos dentro dos resultados podem ser separados por `,`. Assim, vamos canalizar a saída para outro comando `cut`, usando `,` como delimitador.

```
$ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5 | cut -d, -f1  
Dave Edwards  
Emma Jones  
Frank Cassidy  
Grace Kearns  
Henry Adams  
John Chapel
```

Respostas aos Exercícios Exploratórios

- Usando novamente o arquivo `mypasswd` dos exercícios anteriores, imagine um comando Bash que selecione um indivíduo do Main Office para ganhar uma rifa. Use o comando `sed` para imprimir apenas as linhas do Main Office e, em seguida, uma sequência de comando `cut` para recuperar o primeiro nome de cada usuário a partir dessas linhas. Depois, classifique esses nomes aleatoriamente e imprima apenas o nome principal da lista.

Primeiro, explore como o parâmetro `-R` manipula a saída do comando `sort`. Repita esse comando algumas vezes em sua máquina (note que será preciso colocar 'Main Office' entre aspas simples para que o `sed` o trate como uma única string):

```
$ sed -n '/Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R
```

Eis uma solução para o problema:

```
$ sed -n '/Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R | head -1
```

- Quantas pessoas trabalham em Finance, Engineering e Sales? Pense em explorar o comando `uniq`.

Continue incrementando o que você aprendeu nos exercícios anteriores. Tente o seguinte:

```
$ sed -n '/Main Office'/p mypasswd
$ sed -n '/Main Office'/p mypasswd | cut -d, -f2
```

Note que não nos preocupamos com o delimitador `:`. Queremos somente o segundo campo quando dividimos as linhas pelos caracteres `,`.

```
$ sed -n '/Main Office'/p mypasswd | cut -d, -f2 | uniq -c
4 Finance
1 Engineering
2 Sales
```

O comando `uniq` mostra apenas as linhas únicas (não as linhas repetidas) e o parâmetro `-c` diz ao `uniq` para contar as ocorrências de linhas iguais. Mas fique esperto: `uniq` considera apenas as linhas adjacentes. Quando esse não for o caso, será preciso usar o comando `sort`.

3. Agora, vamos preparar um arquivo CSV (valores separados por vírgula) para poder importar facilmente, do arquivo `mypasswd` do exemplo anterior, o arquivo `names.csv` para o LibreOffice. O conteúdo do arquivo terá o seguinte formato:

```
First Name,Last Name,Position
Carol,Smith,Finance
...
John,Chapel,Sales
```

Dica: use os comandos `sed`, `cut` e `paste` para obter os resultados desejados. Note que a vírgula (,) será o delimitador desse arquivo.

Comece com os comandos `sed` e `cut`, incrementando o que aprendemos nos exercícios anteriores:

```
$ sed -n '/Main Office/p mypasswd | cut -d: -f5 | cut -d" " -f1 > firstname
```

Agora temos o arquivo `firstname` com o primeiro nome dos funcionários.

```
$ sed -n '/Main Office/p mypasswd | cut -d: -f5 | cut -d" " -f2 | cut -d, -f1 > lastname
```

Agora temos o arquivo `lastname` com o sobrenome de cada funcionário.

A seguir, vamos determinar o departamento em que cada funcionário trabalha:

```
$ sed -n '/Main Office/p mypasswd | cut -d: -f5 | cut -d, -f2 > department
```

Antes de trabalharmos na resposta final, experimente os seguintes comandos para ver que tipo de saída eles geram:

```
$ cat firstname lastname department
$ paste firstname lastname department
```

E a solução é:

```
$ paste firstname lastname department | tr '\t' ,
```

```
$ paste firstname lastname department | tr '\t' , > names.csv
```

Aqui, usamos o comando `tr` para *traduzir* \t, o separador de tabulação, por um ,. `tr` é bastante útil quando precisamos trocar um caractere por outro. Não deixe de ler as páginas de manual de `tr` e `paste`. Por exemplo, podemos usar a opção `-d` para o delimitador, para tornar o comando anterior menos complexo:

```
$ paste -d, firstname lastname department
```

Usamos o comando `paste` aqui para você se acostumar melhor com ele. No entanto, poderíamos ter executado facilmente todas as tarefas em uma única cadeia de comando:

```
$ sed -n '/Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1,2 | tr ' ' , > names.csv
```

- Suponha que a planilha `names.csv` criada no exercício anterior seja um arquivo importante e queremos ter certeza de que ninguém vai adulterá-lo desde o momento do envio até a recepção pelo destinatário. Como podemos garantir a integridade desse arquivo usando `md5sum`?

Se você consultar as páginas de manual de `md5sum`, `sha256sum` e `sha512sum`, verá que todas começam com o seguinte texto:

“compute and check XXX message digest”

Onde “XXX” é o algoritmo que será usado para criar esse *message digest* (resumo de mensagens).

Usaremos `md5sum` como exemplo e, mais tarde, você poderá tentar com os outros comandos.

```
$ md5sum names.csv
61f0251fcab61d9575b1d0cbf0195e25 names.csv
```

Agora, por exemplo, você pode disponibilizar o arquivo através de um serviço de `ftp` seguro e enviar o *resumo de mensagens* gerado usando outro meio de comunicação seguro. Se o arquivo tiver sido ligeiramente alterado, o *resumo de mensagens* será totalmente diferente. Para comprovar, edite `names.csv` e troque Jones por James, como demonstramos aqui:

```
$ sed -i.backup s/Jones/James/ names.csv
$ md5sum names.csv
```

```
f44a0d68cb480466099021bf6d6d2e65 names.csv
```

Sempre que você disponibilizar arquivos para download, é aconselhável distribuir também um *resumo de mensagens* correspondente para que as pessoas que baixarem aquele arquivo possam produzir um novo *resumo de mensagens* e comparar com o original. Se você visitar o site <https://kernel.org>, encontrará a página <https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/sha256sums.asc>, onde pode obter o sha256sum de todos os arquivos disponíveis para download.

- Você prometeu a si mesmo que leria 100 linhas por dia de um livro clássico e decidiu começar com *Mariner and Mystic* de Herman Melville. Desenvolva um comando usando `split` para dividir este livro em seções de 100 linhas cada. Para obter o livro em formato de texto simples, pesquise em <https://www.gutenberg.org>.

Primeiro, baixamos o livro completo do site do Project Gutenberg, onde você pode obter este e outros livros disponíveis em domínio público.

```
$ wget https://www.gutenberg.org/files/50461/50461-0.txt
```

Pode ser preciso instalar `wget` se ainda não estiver presente no sistema. Outra alternativa é usar `curl`. Use `less` para verificar o livro:

```
$ less 50461-0.txt
```

Agora, vamos dividir o livro em trechos de 100 linhas cada:

```
$ split -l 100 -d 50461-0.txt melville
```

`50461-0.txt` é o arquivo que dividiremos. `melville` será o prefixo dos arquivos divididos. `-l 100` especifica o número de linhas e a opção `-d` diz ao `split` para numerar os arquivos (usando o sufixo fornecido). Podemos usar `nl` em qualquer um dos arquivos divididos (provavelmente não no último) e confirmar que todos eles têm 100 linhas.

- Usando `ls -l` no diretório `/etc`, que tipo de listagem obtemos? Usando o comando `cut` na saída do comando `ls` fornecido, como exibir apenas os nomes dos arquivos? E quanto ao nome dos arquivos e seu proprietário? Junto com os comandos `ls -l` e `cut`, utilize o comando `tr` para *espremer* diversas ocorrências de um espaço em um único espaço para auxiliar na formatação da saída com um comando `cut`.

O comando `ls` sozinho fornece apenas os nomes dos arquivos. Podemos, no entanto, preparar a saída do `ls -l` (a lista longa) para extrair informações mais específicas.

```
$ ls -l /etc | tr -s ' '
drwxr-xr-x,3,root,root,4096,out,24,16:58,acpi
-rw-r--r--,1,root,root,3028,dez,17,2018,adduser.conf
-rw-r--r--,1,root,root,10,out,2,17:38,adjtime
drwxr-xr-x,2,root,root,12288,out,31,09:40,alternatives
-rw-r--r--,1,root,root,401,mai,29,2017,anacrontab
-rw-r--r--,1,root,root,433,out,1,2017,apg.conf
drwxr-xr-x,6,root,root,4096,dez,17,2018,apm
drwxr-xr-x,3,root,root,4096,out,24,16:58,apparmor
drwxr-xr-x,9,root,root,4096,nov,6,20:20,apparmor.d
```

O parâmetro `-s` instrui `tr` a reduzir os espaços repetidos a uma única instância de um espaço. O comando `tr` funciona para qualquer tipo de caractere repetitivo que você especificar. Em seguida, substituímos os espaços por uma vírgula `,`. Na verdade, não precisamos substituir os espaços em nosso exemplo, então apenas omitiremos `,`.

```
$ ls -l /etc | tr -s ' '
drwxr-xr-x 3 root root 4096 out 24 16:58 acpi
-rw-r--r-- 1 root root 3028 dez 17 2018 adduser.conf
-rw-r--r-- 1 root root 10 out 2 17:38 adjtime
drwxr-xr-x 2 root root 12288 out 31 09:40 alternatives
-rw-r--r-- 1 root root 401 mai 29 2017 anacrontab
-rw-r--r-- 1 root root 433 out 1 2017 apg.conf
drwxr-xr-x 6 root root 4096 dez 17 2018 apm
drwxr-xr-x 3 root root 4096 out 24 16:58 apparmor
```

Se quisermos apenas os nomes dos arquivos, só precisamos exibir o nono campo:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9
```

Para o nome de arquivo e seu proprietário, precisamos do nono e do terceiro campos:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9,3
```

E se só precisarmos dos nomes das pastas e seu proprietário?

```
$ ls -l /etc | grep ^d | tr -s ' ' | cut -d" " -f9,3
```

7. Este exercício pressupõe que você está em uma máquina real (não virtual). Também é preciso ter um pendrive à mão. Releia as páginas de manual do comando `tail` e descubra como seguir um arquivo conforme adicionamos texto a ele. Enquanto monitora a saída de um comando `tail` no arquivo `/var/log/syslog`, insira um pendrive. Escreva o comando completo que usaria para obter o Produto (Product), o Fabricante (Manufacturer) e a quantidade total de memória (Blocks) do seu pendrive.

```
$ tail -f /var/log/syslog | grep -i 'product:\|blocks\|manufacturer'
Nov  8 06:01:35 brod-avell kernel: [124954.369361] usb 1-4.3: Product: Cruzer
Blade
Nov  8 06:01:35 brod-avell kernel: [124954.369364] usb 1-4.3: Manufacturer:
SanDisk
Nov  8 06:01:37 brod-avell kernel: [124955.419267] sd 2:0:0:0: [sdc] 61056064
512-byte logical blocks: (31.3 GB/29.1 GiB)
```

Claro que isto é um exemplo e os resultados podem variar dependendo do fabricante do seu pendrive. Observe que agora usamos o parâmetro `-i` com o comando `grep`, pois não sabemos se as strings que procuramos estão em maiúsculas ou minúsculas. Também usamos `|` como um OR (ou) lógico, então procuramos por linhas contendo `product` OR `blocks` OR `manufacturer`.



103.3 Gerenciamento básico de arquivos

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.3

Peso

4

Áreas chave de conhecimento

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

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

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

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



103.3 Lição 1

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

Introdução

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

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

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

Dentre os diferentes tipos de arquivos, temos:

Arquivos regulares

armazenam dados e programas.

Diretórios

contêm outros arquivos.

Arquivos especiais

usados para entrada e saída de dados.

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

Manipulação de arquivos

Usando `ls` para listar arquivos

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

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

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

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

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

O primeiro caractere na saída indica o tipo de arquivo:

-
para um arquivo regular.

d
para um diretório.

c
para um arquivo especial.

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

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

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

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

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

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

```
ls OPTIONS FILE
```

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

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

Criar, copiar, mover e remover arquivos

Criando arquivos com `touch`

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

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

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

```
$ touch file1 file2 file3
```

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

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

```
$ touch -am file3
```

Copiando arquivos com `cp`

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

```
$ cp file1 dir2
```

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

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

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

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

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

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

A sintaxe geral de `cp` é:

```
cp OPTIONS SOURCE DESTINATION
```

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

Movendo arquivos com o `mv`

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

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

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

```
mv FILENAME DESTINATION_DIRECTORY
```

Aqui está um exemplo:

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

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

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

```
$ mv old_file_name new_file_name
```

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

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

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

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

Inversamente, se usarmos `-f`:

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

o arquivo seria sobreescrito à força, sem pedir permissão.

Removendo arquivos com `rm`

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

```
$ rm file1
```

Este comando excluiria file1.

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

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

```
$ rm -f file1
```

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

Podemos excluir vários arquivos ao mesmo tempo:

```
$ rm file1 file2 file3
```

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

A sintaxe de `rm` geralmente é a seguinte:

```
rm OPTIONS FILE
```

Criando e removendo diretórios

Criando diretórios com `mkdir`

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

```
mkdir OPTIONS DIRECTORY_NAME
```

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

```
$ mkdir dir1
```

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

```
$ mkdir dir1 dir2 dir3
```

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

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

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

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

Removendo diretórios com o rmdir

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

```
rmdir OPTIONS DIRECTORY
```

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

```
$ rmdir dir1
```

Este comando excluiria `dir1`.

```
$ rmdir dir1 dir2
```

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

Podemos remover um diretório junto com seu subdiretório:

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

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

Manipulação recursiva de arquivos e diretórios

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

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

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

Listagem recursiva com `ls -R`

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

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

mydirectory/newdirectory:
```

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

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

```
ls -R DIRECTORY_NAME
```

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

```
$ ls -R animal
```

é similar a

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

Cópia recursiva com cp -r

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

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

3 directories, 2 files

```
$ mkdir newcopy
$ cp mydir newcopy
cp: omitting directory 'mydir'
$ cp -r mydir newcopy
* tree newcopy
newcopy
|_mydir
    |_file1
    |_newdir
        |_file2
        |_insidenew
            |_lastdir
```

4 directories, 2 files

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

Para copiar diretórios e subdiretórios, use:

```
cp -r SOURCE DESTINATION
```

Remoção recursiva com `rm -r`

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

WARNING

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

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

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

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

NOTE

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

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

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

O sistema avisa antes de tentar excluir `mydir`.

Globbing de arquivos e caracteres curinga

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

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

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

Veja a seguir alguns exemplos de globbing de arquivos:

`rm *`

Remove todos os arquivos no diretório de trabalho atual.

`ls l?st`

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

`rmdir [a-z]*`

Remove todos os diretórios cujo nome começa com uma letra.

Tipos de caracteres curinga

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

*** (asterisco)**

representa zero, uma ou mais ocorrências de qualquer caractere.

? (interrogação)

representa uma única ocorrência de qualquer caractere.

[](caracteres entre colchetes)

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

O asterisco

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

Por exemplo:

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

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

Da mesma maneira:

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

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

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

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

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

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

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

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

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

```
$ rm *ate*
```

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

O ponto de interrogação

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

Considere a listagem:

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

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

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

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

Da mesma maneira,

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

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

Caracteres entre chaves

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

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

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

Os colchetes também podem indicar intervalos:

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

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

Também podemos definir múltiplos intervalos entre colchetes:

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

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

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

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

Combinando caracteres curinga

Os caracteres curinga podem ser combinados, como em:

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

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

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

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

Exercícios Guiados

1. Considere a listagem abaixo:

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

- O que o caractere d representa na saída?

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

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

2. Considere o comando abaixo:

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

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

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

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

3. Considere a listagem:

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

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

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

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

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

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

Exercícios Exploratórios

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

7. Considere a listagem:

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

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

Resumo

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

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

? (interrogação)

representa uma única ocorrência de qualquer caractere.

[] (caracteres entre colchetes)

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

* (asterisco)

representa zero, uma ou mais ocorrências de qualquer caractere.

Podemos incluir qualquer combinação desses caracteres curinga na mesma instrução.

Respostas aos Exercícios Guiados

1. Considere a listagem abaixo:

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

- O que o caractere d representa na saída?

d é o caractere que identifica um diretório.

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

Por causa da opção -h.

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

Seriam mostrados apenas os nomes dos diretórios e arquivos.

2. Considere o comando abaixo:

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

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

emp_name seria copiado em backup.

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

comando?

-r

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

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

3. Considere a listagem:

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

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

O asterisco *.

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

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

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

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

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

file[0-9].txt

Respostas aos Exercícios Exploratórios

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

```
$ touch dog cat
```

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

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

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

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

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

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

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

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

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

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

7. Considere a listagem:

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

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

```
$ rm *.pdf
```



103.3 Lição 2

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

Introdução

Como encontrar arquivos

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

```
find STARTING_PATH OPTIONS EXPRESSION
```

STARTING_PATH

define o diretório em que a pesquisa se inicia.

OPTIONS

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

EXPRESSION

define os termos da pesquisa.

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

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

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

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

Usando critérios para acelerar a pesquisa

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

As opções para localizar arquivos com base no tipo incluem:

-type f

busca por arquivos.

-type d

busca por diretórios.

-type l

busca por links simbólicos.

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

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

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

-name

pesquisa com base no nome fornecido.

-iname

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

-not

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

-maxdepth N

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

Localizando arquivos por hora de modificação

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

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

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

Localizando arquivos por tamanho

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

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

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

-size 100b

arquivos com exatamente 100 bytes.

-size +100k

arquivos maiores que 100 kilobytes.

-size -20M

arquivos menores que 20 megabytes.

-size +2G

arquivos maiores que 2 gigabytes.

NOTE

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

O que fazer com os resultados

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

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

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

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

Usando o grep para filtrar por arquivos com base no conteúdo

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

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

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

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

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

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

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

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

Arquivos de pacote

O comando `tar` (Arquivamento e compactação)

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

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

A sintaxe do comando `tar` é a seguinte:

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

OPERATION

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

--create (-c)

Cria um novo arquivo tar.

--extract (-x)

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

--list (-t)

Exibe uma lista dos arquivos incluídos no pacote.

OPTIONS

As opções usadas com mais frequência são:

--verbose (-v)

Mostra os arquivos que estão sendo processados pelo comando tar.

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

Especifica o nome de arquivo do pacote.

ARCHIVE_NAME

O nome do arquivo de pacote.

FILE_NAME(S)

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

Criando um arquivo de pacote

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

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

Eis o que essas opções significam de fato:

-c

Cria um arquivo de pacote.

-v

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

-f

Permite especificar o nome de arquivo do pacote.

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

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

NOTE

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

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

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

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

Extraindo um pacote

Podemos extrair um arquivo de pacote usando o tar:

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

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

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

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

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

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

```
$ ls /tmp
```

```
stuff
```

Compactando com o tar

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

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

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

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

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

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

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

```
$ tar -xzvf archive.tar.gz
```

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

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

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

gzip

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

gzip

remove os arquivos originais após criar o arquivo compactado.

O comando `bzip2` funciona de maneira semelhante.

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

O comando cpio

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

O `cpio` executa as seguintes operações:

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

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

Para criar um arquivo `cpio`, usamos:

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

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

Para extrair o arquivo de pacote, usamos:

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

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

O comando dd

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

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

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

NOTE

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

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

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

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

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

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

Exercícios Guiados

1. Considere a listagem a seguir:

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

- Que tipo de arquivos esse comando produziria?

- A busca começaria em qual diretório?

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

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

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

Exercícios Exploratórios

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

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

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

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

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

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

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

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

Resumo

Nesta seção, você aprendeu:

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

Respostas aos Exercícios Guiados

1. Considere a listagem a seguir:

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

- Que tipo de arquivos esse comando produziria?

Diretórios.

- A busca começaria em qual diretório?

/home/frank/Documents

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

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

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

A opção `-z`.

Respostas aos Exercícios Exploratórios

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

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

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

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

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

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

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

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

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

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

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

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

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

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



103.4 Fluxos, pipes (canalização) e redirecionamentos de saída

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.4

Peso

4

Áreas chave de conhecimento

- Redirecionamento da entrada padrão, da saída padrão e dos erros padrão.
- Canalização (piping) da saída de um comando à entrada de outro comando.
- Usar a saída de um comando como argumento para outro comando.
- Enviar a saída de um comando simultaneamente para a saída padrão e um arquivo.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- tee
- xargs



103.4 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.4 Usando fluxos, pipes e redirecionamentos
Lição:	1 de 2

Introdução

Todos os programas de computador seguem o mesmo princípio geral: os dados recebidos de alguma fonte são transformados para gerar um resultado inteligível. No contexto do shell do Linux, a fonte de dados pode ser um arquivo local, um arquivo remoto, um dispositivo (como um teclado), etc. A saída do programa geralmente é exibida em uma tela, mas também é comum armazenar os dados de saída em um sistema de arquivos local, enviar para um dispositivo remoto, reproduzi-lo em alto-falantes de áudio, etc.

Os sistemas operacionais inspirados no Unix, como o Linux, oferecem uma grande variedade de métodos de entrada/saída. Em particular, o método dos *descritores de arquivo* permite associar dinamicamente números inteiros a canais de dados, para que um processo possa referenciá-los como seus fluxos de dados de entrada/saída.

Os processos padrão do Linux têm três canais de comunicação abertos por padrão: o canal de *entrada padrão* (na maioria das vezes simplesmente chamado de *stdin*), o canal de *saída padrão* (*stdout*) e o canal de *erro padrão* (*stderr*). Os descritores numéricos de arquivo atribuídos a esses canais são `0` para *stdin*, `1` para *stdout* e `2` para *stderr*. Os canais de comunicação também são acessíveis por meio dos dispositivos especiais `/dev/stdin`, `/dev/stdout` e `/dev/stderr`.

Esses três canais de comunicação permitem que os programadores escrevam códigos que leem e gravam dados sem se preocupar com o tipo de mídia de onde vêm ou para o qual vão. Por exemplo, se um programa precisa de um conjunto de dados como entrada, pode simplesmente solicitar os dados da entrada padrão; será fornecido aquilo que estiver sendo usado como entrada padrão. Da mesma forma, o método mais simples que um programa pode usar para exibir sua saída é escrevê-la na saída padrão. Em uma sessão comum do shell, o teclado é definido como stdin e a tela do monitor como stdout e stderr.

O shell Bash tem a capacidade de reatribuir os canais de comunicação ao carregar um programa. Ele permite, por exemplo, substituir a tela como a saída padrão e usar um arquivo no sistema de arquivos local como stdout.

Redirecionamentos

A reatribuição do descritor de arquivo de um canal no ambiente shell é chamada de *redirecionamento*. Um redirecionamento é definido por um caractere especial na linha de comando. Por exemplo, para redirecionar a saída padrão de um processo para um arquivo, o símbolo de *maior que* > é posicionado no final do comando e seguido pelo caminho até o arquivo que receberá a saída redirecionada:

```
$ cat /proc/cpuinfo >/tmp/cpu.txt
```

Por padrão, apenas o conteúdo que chega a stdout é redirecionado. Isso ocorre porque o valor numérico do descritor de arquivo deve ser especificado logo antes do símbolo de maior que e, quando não especificado, o Bash redireciona a saída padrão. Portanto, usar > é equivalente a usar 1> (o valor do descritor de arquivo de stdout é 1).

Para capturar o conteúdo de stderr, o redirecionamento 2> deve ser usado. A maioria dos programas de linha de comando enviam informações de depuração e mensagens de erro para o canal de erro padrão. É possível, por exemplo, capturar a mensagem de erro gerada por uma tentativa de leitura de um arquivo inexistente:

```
$ cat /proc/cpu_info 2>/tmp/error.txt
$ cat /tmp/error.txt
cat: /proc/cpu_info: No such file or directory
```

Tanto stdout quanto stderr são redirecionados para o mesmo destino com &> ou >&. É importante não colocar nenhum espaço ao lado do "e" comercial, caso contrário o Bash o interpretará como uma instrução para executar o processo em segundo plano e não para executar o redirecionamento.

O destino deve ser um caminho para um arquivo gravável, como /tmp/cpu.txt, ou um descritor de

arquivo gravável. O destino de um descritor de arquivo é representado por um "e" comercial seguido pelo valor numérico do descritor de arquivo. Por exemplo, `1>&2` redireciona stdout para stderr. Para fazer o oposto, stderr para stdout, devemos usar `2>&1`.

Embora não seja muito útil, visto que existe uma maneira mais curta de executar a mesma tarefa, é possível redirecionar stderr para stdout e, em seguida, redirecioná-lo para um arquivo. Por exemplo, um redirecionamento para gravar stderr e stdout em um arquivo chamado `log.txt` pode ser escrito como `>log.txt 2>&1`. No entanto, o principal motivo para redirecionar stderr para stdout é permitir a análise de mensagens de depuração e erro. É possível redirecionar a saída padrão de um programa para a entrada padrão de outro programa, mas não é possível redirecionar diretamente o erro padrão para a entrada padrão de outro programa. Assim, as mensagens do programa enviadas para stderr primeiro precisam ser redirecionadas para stdout a fim de serem lidas pelo stdin de outro programa.

Para simplesmente descartar a saída de um comando, seu conteúdo pode ser redirecionado para o arquivo especial `/dev/null`. Por exemplo, `>log.txt 2>/dev/null` salva o conteúdo de stdout no arquivo `log.txt` e descarta o stderr. O arquivo `/dev/null` pode ser escrito por qualquer usuário, mas nenhum dado pode ser recuperado dele, pois não é armazenado em lugar nenhum.

Uma mensagem de erro é apresentada se o destino especificado não for gravável (se o caminho apontar para um diretório ou um arquivo somente leitura) e nenhuma modificação for feita no destino. No entanto, um redirecionamento de saída sobrescreve um destino gravável existente sem pedir nenhuma confirmação. Os arquivos são substituídos pelos redirecionamentos de saída, a menos que a opção `noclobber` esteja habilitada no Bash, o que pode ser feito para a sessão atual com o comando `set -o noclobber` ou `set -C`:

```
$ set -o noclobber
$ cat /proc/cpu_info 2>/tmp/error.txt
-bash: /tmp/error.txt: cannot overwrite existing file
```

Para remover a opção `noclobber` da sessão atual, execute `set +o noclobber` ou `set +C`. Para tornar a opção `noclobber` persistente, ela deve ser incluída no perfil Bash do usuário ou no perfil de todo o sistema.

Mesmo com a opção `noclobber` habilitada, é possível anexar dados redirecionados ao conteúdo existente. Usamos para isso um redirecionamento escrito com dois símbolos de maior que, `>>`:

```
$ cat /proc/cpu_info 2>>/tmp/error.txt
$ cat /tmp/error.txt
cat: /proc/cpu_info: No such file or directory
cat: /proc/cpu_info: No such file or directory
```

No exemplo anterior, a nova mensagem de erro foi anexada à existente no arquivo `/tmp/error.txt`. Se o arquivo ainda não existir, ele será criado com os novos dados.

A fonte de dados da entrada padrão de um processo também pode ser reatribuída. O símbolo de menor que < é usado para redirecionar o conteúdo de um arquivo para o stdin de um processo. Nesse caso, os dados fluem da direita para a esquerda: o descritor reatribuído é considerado como sendo 0 à esquerda do símbolo de menor que e a fonte de dados (um caminho para um arquivo) deve estar à direita do símbolo de menor que. O comando `uniq`, como a maioria dos utilitários de linha de comando para processamento de texto, aceita os dados enviados para stdin por padrão:

```
$ uniq -c </tmp/error.txt
 2 cat: /proc/cpu_info: No such file or directory
```

A opção `-c` faz com que o `uniq` exiba quantas vezes uma linha repetida aparece no texto. Como o valor numérico do descritor de arquivo redirecionado foi suprimido, o comando de exemplo é equivalente a `uniq -c 0</tmp/error.txt`. O uso de um descritor de arquivo diferente de 0 em um redirecionamento de entrada só faz sentido em determinados contextos, porque um programa pode possivelmente solicitar dados dos descritores 3, 4, etc. De fato, os programas podem usar qualquer número inteiro maior que 2 como novos descritores de arquivo para entrada/saída de dados. Por exemplo, o código em C a seguir lê dados do descritor de arquivo 3 e simplesmente os reproduz para o descritor 4:

NOTE

O programa deve gerenciar corretamente esses descritores de arquivo, caso contrário ele pode tentar uma operação inválida de leitura ou gravação e travar.

```
#include <stdio.h>

int main(int argc, char **argv){
    FILE *fd_3, *fd_4;
    // Open file descriptor 3
    fd_3 = fdopen(3, "r");
    // Open file descriptor 4
    fd_4 = fdopen(4, "w");
    // Read from file descriptor 3
    char buf[32];
    while ( fgets(buf, 32, fd_3) != NULL ){
        // Write to file descriptor 4
        fprintf(fd_4, "%s", buf);
    }
    // Close both file descriptors
    fclose(fd_3);
```

```
fclose(fd_4);
}
```

Para testá-lo, salve o código de amostra como `fd.c` e compile-o com `gcc -o fd fd.c`. Este programa precisa que os descritores de arquivo 3 e 4 estejam disponíveis para poder ler e gravar neles. Como exemplo, o arquivo `/tmp/error.txt` criado anteriormente pode ser usado como fonte para o descritor de arquivo 3 e o descritor de arquivo 4 pode ser redirecionado para `stdout`:

```
$ ./fd 3</tmp/error.txt 4>&1
cat: /proc/cpu_info: No such file or directory
cat: /proc/cpu_info: No such file or directory
```

Do ponto de vista do programador, o uso de descritores de arquivo evita a obrigação de lidar com a análise de opções (parsing) e com os caminhos do sistema de arquivos. É possível até usar o mesmo descritor de arquivo como entrada e saída. Nesse caso, o descritor de arquivo é definido na linha de comando com os símbolos menor que e maior que, como em `3<>/tmp/error.txt`.

Here Document e Here String

Outra forma de redirecionar a entrada envolve os métodos *Here document* e *Here string*. O redirecionamento *Here document* permite digitar um texto de várias linhas que será usado como o conteúdo redirecionado. Dois símbolos de menor que `<<` indicam um redirecionamento de *Here document*:

```
$ wc -c <<EOF
> How many characters
> in this Here document?
> EOF
43
```

À direita dos dois símbolos de menor que `<<` está o termo de fim `EOF`. O modo de inserção termina assim que for inserida uma linha contendo apenas o termo de fim. Qualquer outro termo pode ser usado como termo de fim, mas é importante não colocar caracteres em branco entre o símbolo de menor que e o termo de fim. No exemplo acima, as duas linhas de texto foram enviadas para o `stdin` do comando `wc -c`, que exibe a contagem de caracteres. Assim como acontece com os redirecionamentos de entrada para arquivos, o `stdin` (descritor de arquivo 0) é pressuposto se o descritor de arquivo redirecionado for suprimido.

O método de *Here string* é muito parecido com o método de *Here document*, mas para uma linha apenas:

```
$ wc -c <<<"How many characters in this Here string?"  
41
```

Neste exemplo, a string à direita dos três sinais de menor é enviada para o stdin de `wc -c`, que conta o número de caracteres. As strings contendo espaços devem estar entre aspas, senão apenas a primeira palavra será usada como Here string e as restantes serão passadas como argumentos para o comando.

Exercícios Guiados

- Além dos arquivos de texto, o comando `cat` também pode trabalhar com dados binários, como enviar o conteúdo de um dispositivo de bloco para um arquivo. Usando redirecionamento, como o `cat` pode enviar o conteúdo do dispositivo `/dev/sdc` para o arquivo `sdc.img` no diretório atual?

- Qual é o nome do canal padrão redirecionado pelo comando `date 1> now.txt`?

- Ao tentar sobrescrever um arquivo usando redirecionamento, um usuário recebe uma mensagem de erro informando que a opção `noclobber` está habilitada. Como essa opção pode ser desativada para a sessão atual?

- Qual será o resultado do comando `cat <<.>/dev/stdout`?

Exercícios Exploratórios

1. O comando `cat /proc/cpu_info` exibe uma mensagem de erro porque `/proc/cpu_info` não existe. Para onde o comando `cat /proc/cpu_info 2>1` redireciona a mensagem de erro?

2. Ainda será possível descartar o conteúdo enviado para `/dev/null` se a opção `noclobber` estiver habilitada para a sessão de shell atual?

3. Sem usar `echo`, como o conteúdo da variável `$USER` poderia ser redirecionado para o `stdin` do comando `sha1sum`?

4. O kernel do Linux mantém links simbólicos em `/proc/PID/fd/` para cada arquivo aberto por um processo, onde `PID` é o número de identificação do processo correspondente. Como o administrador do sistema poderia usar esse diretório para verificar a localização dos arquivos de log abertos pelo `nginx`, supondo que seu `PID` é `1234`?

5. É possível fazer cálculos aritméticos usando apenas comandos internos do shell, mas cálculos de ponto flutuante requerem programas específicos, como o `bc` (*basic calculator*). Com o `bc` é possível até mesmo especificar o número de casas decimais, com o parâmetro `escala`. No entanto, o `bc` aceita operações apenas por meio de sua entrada padrão, geralmente inserida no modo interativo. Usando uma Here string, como a operação de ponto flutuante `scale=6; 1/3` pode ser enviada para a entrada padrão de `bc`?

Resumo

Esta lição cobre métodos para executar um programa redirecionando seus canais de comunicação padrão. Os processos do Linux usam esses canais padrão como *descritores de arquivo* genéricos para ler e gravar dados, tornando possível transferi-los arbitrariamente para arquivos ou dispositivos. A lição demonstra as seguintes etapas:

- O que são descritores de arquivos e qual seu papel no Linux.
- Os canais de comunicação padrão em todos os processos: *stdin*, *stdout* e *stderr*.
- Como executar corretamente um comando usando redirecionamento de dados, tanto na entrada quanto na saída.
- Como usar *Here Documents* e *Here Strings* nos redirecionamentos de entrada.

Os comandos e procedimentos abordados foram:

- Operadores de redirecionamento: `>`, `<`, `>>`, `<<`, `<<<`.
- Comandos `cat`, `set`, `uniq` e `wc`.

Respostas aos Exercícios Guiados

- Além dos arquivos de texto, o comando `cat` também pode trabalhar com dados binários, como enviar o conteúdo de um dispositivo de bloco para um arquivo. Usando redirecionamento, como o `cat` pode enviar o conteúdo do dispositivo `/dev/sdc` para o arquivo `sdc.img` no diretório atual?

```
$ cat /dev/sdc > sdc.img
```

- Qual é o nome do canal padrão redirecionado pelo comando `date 1> now.txt`?

Standard output ou stdout

- Ao tentar sobrescrever um arquivo usando redirecionamento, um usuário recebe uma mensagem de erro informando que a opção `noclobber` está habilitada. Como essa opção pode ser desativada para a sessão atual?

```
set +C ou set +o noclobber
```

- Qual será o resultado do comando `cat <<.>/dev/stdout`?

O Bash entrará no modo de entrada Heredoc e sairá quando um ponto final aparecer sozinho em uma linha. O texto digitado será redirecionado para stdout (impresso na tela).

Respostas aos Exercícios Exploratórios

1. O comando `cat /proc/cpu_info` exibe uma mensagem de erro porque `/proc/cpu_info` não existe. Para onde o comando `cat /proc/cpu_info 2>1` redireciona a mensagem de erro?

Para um arquivo chamado `1` no diretório atual.

2. Ainda será possível descartar o conteúdo enviado para `/dev/null` se a opção `noclobber` estiver habilitada para a sessão de shell atual?

Sim. `/dev/null` é um arquivo especial, não afetado por `noclobber`.

3. Sem usar `echo`, como o conteúdo da variável `$USER` poderia ser redirecionado para o `stdin` do comando `sha1sum`?

```
$ sha1sum <<<$USER
```

4. O kernel do Linux mantém links simbólicos em `/proc/PID/fd/` para cada arquivo aberto por um processo, onde `PID` é o número de identificação do processo correspondente. Como o administrador do sistema poderia usar esse diretório para verificar a localização dos arquivos de log abertos pelo `nginx`, supondo que seu `PID` é `1234`?

Emitindo o comando `ls -l /proc/1234/fd`, que exibirá os destinos de cada link simbólico no diretório.

5. É possível fazer cálculos aritméticos usando apenas comandos internos do shell, mas cálculos de ponto flutuante requerem programas específicos, como o `bc` (*basic calculator*). Com o `bc` é possível até mesmo especificar o número de casas decimais, com o parâmetro `escala`. No entanto, o `bc` aceita operações apenas por meio de sua entrada padrão, geralmente inserida no modo interativo. Usando uma Here string, como a operação de ponto flutuante `scale=6; 1/3` pode ser enviada para a entrada padrão de `bc`?

```
$ bc <<<"scale=6; 1/3"
```



103.4 Lição 2

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.4 Usando fluxos, pipes e redirecionamentos
Lição:	2 de 2

Introdução

Um aspecto da filosofia Unix afirma que cada programa precisa ter um propósito específico e não deve tentar incorporar recursos fora de seu escopo. Mas manter as coisas simples não significa que os resultados serão menos elaborados, já que diferentes programas podem ser encadeados para produzir uma saída combinada. O caractere de barra vertical `|`, também conhecido como símbolo *pipe*, pode ser usado para criar uma canalização (pipeline) conectando a saída de um programa diretamente à entrada de outro programa, ao passo que a *substituição de comandos* permite armazenar a saída de um programa em uma variável ou usá-lo diretamente como argumento para outro comando.

Pipes

Ao contrário dos redirecionamentos, com os pipes os dados fluem da esquerda para a direita na linha de comando e o destino é outro processo, não um caminho do sistema de arquivos, descriptor de arquivo ou Here document. O caractere de barra vertical `|` manda o shell iniciar todos os comandos distintos ao mesmo tempo e conectar a saída do comando anterior à entrada do comando seguinte, da esquerda para a direita. Por exemplo, em vez de usar redirecionamentos, o conteúdo do arquivo `/proc/cpuinfo` enviado para a saída padrão por `cat` pode ser canalizado para o `stdin` de `wc` com o

seguinte comando:

```
$ cat /proc/cpuinfo | wc  
208      1184      6096
```

Na ausência do caminho para um arquivo, `wc` conta o número de linhas, palavras e caracteres que recebe em seu `stdin`, como é o caso no exemplo. Muitos pipes podem estar presentes em um comando composto. No exemplo a seguir, dois pipes são usados:

```
$ cat /proc/cpuinfo | grep 'model name' | uniq  
model name      : Intel(R) Xeon(R) CPU           X5355 @ 2.66GHz
```

O conteúdo do arquivo `/proc/cpuinfo` produzido por `cat /proc/cpuinfo` foi canalizado para o comando `grep 'model name'`, que em seguida seleciona apenas as linhas contendo o termo `model name`. A máquina que executa o exemplo tem muitas CPUs, portanto existem linhas repetidas com `model name`. O último pipe conecta `grep 'model name'` ao `uniq`, que é responsável por pular qualquer linha idêntica à anterior.

Os pipes podem ser combinados com redirecionamentos na mesma linha de comando. O exemplo anterior pode ser reescrito em uma forma mais simples:

```
$ grep 'model name' </proc/cpuinfo | uniq  
model name      : Intel(R) Xeon(R) CPU           X5355 @ 2.66GHz
```

O redirecionamento de entrada para `grep` não é estritamente necessário, pois `grep` aceita um caminho de arquivo como argumento, mas o exemplo demonstra como construir comandos combinados.

Pipes e redirecionamentos são exclusivos, ou seja, uma origem pode ser mapeada para apenas um destino. Ainda assim, é possível redirecionar uma saída para um arquivo e ainda vê-la na tela com o programa `tee`. Para isso, o primeiro programa envia sua saída para o `stdin` de `tee` e um nome de arquivo é fornecido a este último para armazenar os dados:

```
$ grep 'model name' </proc/cpuinfo | uniq | tee cpu_model.txt  
model name      : Intel(R) Xeon(R) CPU           X5355 @ 2.66GHz  
$ cat cpu_model.txt  
model name      : Intel(R) Xeon(R) CPU           X5355 @ 2.66GHz
```

A saída do último programa na cadeia, gerada por `uniq`, é exibida e armazenada no arquivo `cpu_model.txt`. Para não sobreescriver o conteúdo do arquivo fornecido, e sim anexar dados a ele, a opção `-a` deve ser fornecida para `tee`.

Apenas a saída padrão de um processo é capturada por um pipe. Digamos que você precise passar por um longo processo de compilação na tela e, ao mesmo tempo, salvar tanto a saída padrão quanto o erro padrão em um arquivo para inspeção posterior. Supondo que seu diretório atual não tenha um `Makefile`, o seguinte comando gerará um erro:

```
$ make | tee log.txt
make: *** No targets specified and no makefile found. Stop.
```

Embora exibida na tela, a mensagem de erro gerada por `make` não foi capturada por `tee` e o arquivo `log.txt` foi criado vazio. É preciso fazer um redirecionamento antes que um pipe possa capturar o `stderr`:

```
$ make 2>&1 | tee log.txt
make: *** No targets specified and no makefile found. Stop.
$ cat log.txt
make: *** No targets specified and no makefile found. Stop.
```

Neste exemplo, o `stderr` de `make` foi redirecionado para o `stdout`, de forma que o `tee` foi capaz de capturá-lo com um pipe, exibi-lo na tela e salvá-lo no arquivo `log.txt`. Em casos como esse, pode ser útil salvar as mensagens de erro para inspeção posterior.

Substituição de comando

Outro método para capturar a saída de um comando é a *substituição de comando*. Ao colocar um comando entre crases, o Bash o substitui por sua saída padrão. O exemplo a seguir mostra como usar o `stdout` de um programa como argumento para outro programa:

```
$ mkdir `date +%Y-%m-%d`
$ ls
2019-09-05
```

A saída do programa `date`, a data atual formatada como *ano-mês-dia*, foi usada como um argumento para criar um diretório com o `mkdir`. Um resultado idêntico é obtido usando `$()` em vez de crases:

```
$ rmdir 2019-09-05
```

```
$ mkdir $(date +%Y-%m-%d)
$ ls
2019-09-05
```

O mesmo método pode ser usado para armazenar a saída de um comando como uma variável:

```
$ OS=`uname -o`
$ echo $OS
GNU/Linux
```

O comando `uname -o` retorna o nome genérico do sistema operacional atual, que foi armazenado na variável de sessão `OS`. Atribuir a saída de um comando a uma variável é muito útil em scripts, possibilitando armazenar e avaliar os dados de várias maneiras distintas.

Dependendo da saída gerada pelo comando substituído, a substituição do comando interno do shell pode não ser apropriada. Um método mais sofisticado para usar a saída de um programa como argumento de outro programa emprega um intermediário chamado `xargs`. O programa `xargs` usa o conteúdo que recebe via `stdin` para executar um determinado comando com o conteúdo como argumento. O exemplo a seguir mostra o `xargs` executando o programa `identify` com argumentos fornecidos pelo programa `find`:

```
$ find /usr/share/icons -name 'debian*' | xargs identify -format "%f: %wx%h\n"
debian-swirl.svg: 48x48
debian-swirl.png: 22x22
debian-swirl.png: 32x32
debian-swirl.png: 256x256
debian-swirl.png: 48x48
debian-swirl.png: 16x16
debian-swirl.png: 24x24
debian-swirl.svg: 48x48
```

O programa `identify` é parte do *ImageMagick*, um conjunto de ferramentas de linha de comando para inspecionar, converter e editar a maioria dos tipos de arquivo de imagem. No exemplo, o `xargs` pegou todos os caminhos listados por `find` e os colocou como argumentos para `identify`, que então exibe as informações para cada arquivo formatado conforme exigido pela opção `-format`. Os arquivos encontrados pelo `find` no exemplo são imagens contendo o logotipo da distribuição em um sistema de arquivos Debian. `-format` é um parâmetro para `identify`, não para `xargs`.

A opção `-n 1` exige que o `xargs` execute o comando fornecido com apenas um argumento por vez. No caso do exemplo, em vez de passar todos os caminhos encontrados por `find` como uma lista de

argumentos para `identify`, o uso de `xargs -n 1` executaria o comando `identify` para cada caminho separadamente. Usar `-n 2` executaria o `identify` com dois caminhos como argumentos, `-n 3` com três caminhos como argumentos e assim por diante. Da mesma forma, quando o `xargs` processa conteúdos com várias linhas – como é o caso com a entrada fornecida por `find` – a opção `-L` pode ser usada para limitar quantas linhas serão usadas como argumentos por execução do comando.

NOTE

Pode ser desnecessário usar o `xargs` com a opção `-n 1` ou `-L 1` para processar a saída gerada pelo `find`. O comando `find` tem a opção `-exec` para executar um comando determinado para cada item do resultado da busca.

Se os caminhos contiverem caracteres de espaço, é importante executar o `find` com a opção `-print0`. Esta opção instrui o `find` a usar um caractere nulo entre cada entrada para que a lista possa ser analisada corretamente por `xargs` (a saída foi suprimida):

```
$ find . -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -print0 |
xargs -0 du | sort -n
```

A opção `-0` diz ao `xargs` que o caractere nulo deve ser usado como separador. Dessa forma, os caminhos de arquivo fornecidos pelo `find` são analisados corretamente, mesmo se contiverem espaços em branco ou outros caracteres especiais. O exemplo anterior mostra como usar o comando `du` para descobrir o uso de espaço em disco por cada arquivo encontrado e em seguida classificar os resultados por tamanho. A saída foi suprimida para fins de concisão. Observe que para cada critério de pesquisa é necessário incluir a opção `-print0` para `find`.

Por padrão, o `xargs` coloca por último os argumentos do comando executado. Para mudar esse comportamento, usamos a opção `-I`:

```
$ find . -mindepth 2 -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -
-print0 | xargs -0 -I PATH mv PATH ./
```

No último exemplo, todo arquivo encontrado por `find` é movido para o diretório atual. Como o(s) caminho(s) de origem devem ser informados para o `mv` antes do caminho de destino, um termo de substituição é dado à opção `-I` do `xargs`, que é então apropriadamente colocado junto a `mv`. Ao usar o caractere nulo como separador, não é necessário colocar o termo de substituição entre aspas.

Exercícios Guiados

1. É conveniente salvar a data de execução das ações realizadas por scripts automáticos. O comando `date +%Y-%m-%d` mostra a data atual no formato *ano-mês-dia*. Como a saída desse comando pode ser armazenada em uma variável do shell chamada `TODAY` usando a substituição de comando?

2. Usando o comando `echo`, como o conteúdo da variável `TODAY` pode ser enviado para a saída padrão do comando `sed s/-/.g`?

3. Como a saída do comando `date +%Y-%m-%d` pode ser usada como uma Here string para o comando `sed s/-/.g`?

4. O comando `convert image.jpeg -resize 25% small/image.jpeg` cria uma versão menor de `image.jpeg` e coloca a imagem resultante em um arquivo com o mesmo nome dentro do subdiretório `small`. Usando o `xargs`, como é possível executar o mesmo comando para todas as imagens listadas no arquivo `filelist.txt`?

Exercícios Exploratórios

- Uma rotina de backup simples cria periodicamente uma imagem da partição /dev/sda1 com `dd < /dev/sda1 > sda1.img`. Para realizar futuras verificações de integridade de dados, a rotina também gera um hash SHA1 do arquivo com `sha1sum < sda1.img > sda1.sha1`. Adicionando pipes e o comando `tee`, como esses dois comandos poderiam ser combinados em um só?

- O comando `tar` é usado para empacotar muitos arquivos em um só, preservando a estrutura de diretórios. A opção `-T` permite especificar um arquivo contendo os caminhos a arquivar. Por exemplo, `find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T` – cria o arquivo tar compactado `etc.tar.xz` a partir da lista fornecida pelo comando `find` (a opção `-T` – indica a entrada padrão como a lista de caminhos). Para evitar possíveis erros de análise devido a caminhos que contêm espaços, quais opções deveriam estar presentes para os comandos `find` e `tar`?

- Em vez de abrir uma nova sessão remota do shell, o comando `ssh` pode simplesmente executar um comando indicado como argumento: `ssh user@storage "remote command"`. Dado que `ssh` também permite redirecionar a saída padrão de um programa local para a entrada padrão do programa remoto, como o comando `cat` canalizaria um arquivo local chamado `etc.tar.gz` para `/srv/backup/etc.tar.gz` em `user@storage` através de `ssh`?

Resumo

Esta lição cobre as técnicas tradicionais de comunicação entre processos empregadas pelo Linux. A *canalização de comandos* cria um canal de comunicação unilateral entre dois processos e a *substituição de comandos* permite armazenar a saída de um processo em uma variável shell. A lição passa pelas seguintes etapas:

- Como *pipes* podem ser usados para transmitir a saída de um processo para a entrada de outro processo.
- A finalidade dos comandos `tee` e `xargs`.
- Como capturar a saída de um processo com a *substituição de comando*, armazenando-a em uma variável ou usando-a diretamente como parâmetro para outro comando.

Os comandos e procedimentos abordados foram:

- Canalização de comandos com `|`.
- Substituição de comandos com crases e `$()`.
- Os comandos `tee`, `xargs` e `find`.

Respostas aos Exercícios Guiados

- É conveniente salvar a data de execução das ações realizadas por scripts automáticos. O comando `date +%Y-%m-%d` mostra a data atual no formato *ano-mês-dia*. Como a saída desse comando pode ser armazenada em uma variável do shell chamada `TODAY` usando a substituição de comando?

```
$ TODAY=`date +%Y-%m-%d`
```

ou

```
$ TODAY=$(date +%Y-%m-%d)
```

- Usando o comando `echo`, como o conteúdo da variável `TODAY` pode ser enviado para a saída padrão do comando `sed s/-./g`?

```
$ echo $TODAY | sed s/-./g
```

- Como a saída do comando `date +%Y-%m-%d` pode ser usada como uma Here string para o comando `sed s/-./g`?

```
$ sed s/-./g <<< `date +%Y-%m-%d`
```

ou

```
$ sed s/-./g <<< $(date +%Y-%m-%d)
```

- O comando `convert image.jpeg -resize 25% small/image.jpeg` cria uma versão menor de `image.jpeg` e coloca a imagem resultante em um arquivo com o mesmo nome dentro do subdiretório `small`. Usando o `xargs`, como é possível executar o mesmo comando para todas as imagens listadas no arquivo `filelist.txt`?

```
$ xargs -I IMG convert IMG -resize 25% small/IMG < filelist.txt
```

ou

```
$ cat filelist.txt | xargs -I IMG convert IMG -resize 25% small/IMG
```

Respostas aos Exercícios Exploratórios

1. Uma rotina de backup simples cria periodicamente uma imagem da partição /dev/sda1 com `dd < /dev/sda1 > sda1.img`. Para realizar futuras verificações de integridade de dados, a rotina também gera um hash SHA1 do arquivo com `sha1sum < sda1.img > sda1.sha1`. Adicionando pipes e o comando `tee`, como esses dois comandos poderiam ser combinados em um só?

```
# dd < /dev/sda1 | tee sda1.img | sha1sum > sda1.sha1
```

2. O comando `tar` é usado para empacotar muitos arquivos em um só, preservando a estrutura de diretórios. A opção `-T` permite especificar um arquivo contendo os caminhos a arquivar. Por exemplo, `find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T` – cria o arquivo tar compactado `etc.tar.xz` a partir da lista fornecida pelo comando `find` (a opção `-T` – indica a entrada padrão como a lista de caminhos). Para evitar possíveis erros de análise devido a caminhos que contêm espaços, quais opções deveriam estar presentes para os comandos `find` e `tar`?

Options `-print0` and `--null`:

```
$ find /etc -type f -print0 | tar -cJ -f /srv/backup/etc.tar.xz --null -T -
```

3. Em vez de abrir uma nova sessão remota do shell, o comando `ssh` pode simplesmente executar um comando indicado como argumento: `ssh user@storage "remote command"`. Dado que `ssh` também permite redirecionar a saída padrão de um programa local para a entrada padrão do programa remoto, como o comando `cat` canalizaria um arquivo local chamado `etc.tar.gz` para `/srv/backup/etc.tar.gz` em `user@storage` através de `ssh`?

```
$ cat etc.tar.gz | ssh user@storage "cat > /srv/backup/etc.tar.gz"
```

or

```
$ ssh user@storage "cat > /srv/backup/etc.tar.gz" < etc.tar.gz
```



103.5 Criar, monitorar e finalizar processos

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.5

Peso

4

Áreas chave de conhecimento

- Executar processos em primeiro e segundo plano.
- Marcar um programa para que continue a rodar depois do logout.
- Monitorar processos ativos.
- Selecionar e ordenar processos para serem exibidos.
- Enviar sinais para os processos.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- &
- bg
- fg
- jobs
- kill
- nohup
- ps
- top
- free

- `uptime`
- `pgrep`
- `pkill`
- `killall`
- `watch`
- `screen`
- `tmux`



103.5 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.5 Criar, monitorar e eliminar processos
Lição:	1 de 2

Introdução

A cada vez que invocamos um comando, um ou mais processos são iniciados. Um administrador de sistema experiente não só precisa criar processos, mas também ser capaz de controlá-los e enviar diferentes tipos de sinais a eles se e quando necessário. Nesta lição, discorreremos sobre o controle de trabalhos e o monitoramento de processos.

Controle de jobs

Jobs (trabalhos) são processos iniciados de forma interativa através de um terminal, enviados para o segundo plano e ainda não finalizados. Para descobrir mais sobre os jobs ativos (e seus status) em seu sistema Linux, execute `jobs`:

```
$ jobs
```

O comando `jobs` acima não produziu nenhuma saída, o que significa que não há jobs ativos no momento. Vamos criar nosso primeiro job executando um comando que leva algum tempo para terminar de ser executado (o comando `sleep` com um parâmetro de `60`) e—durante a execução—pressionamos `Ctrl + Z`:

```
$ sleep 60
^Z
[1]+ Stopped                 sleep 60
```

A execução do comando foi interrompida (ou, mais exatamente, suspensa) e o prompt de comando está disponível novamente. Se você procurar por jobs uma segunda vez, encontrará aquele que foi *suspensão*:

```
$ jobs
[1]+ Stopped                 sleep 60
```

Vamos entender melhor essa saída:

[1]

Este número é o identificador do trabalho e pode ser usado --precedido por um símbolo de porcentagem (%)—para alterar o status do job com os utilitários `fg`, `bg` e `kill` (como ensinaremos mais tarde).

+

O sinal de mais indica o job atual por padrão (ou seja, o último a ser suspenso ou mandado ao segundo plano). O job anterior é sinalizado com um sinal de menos (-). Quaisquer outros jobs anteriores não são sinalizados.

Stopped

Descrição do status do job.

sleep 60

O comando ou job em si.

Com a opção `-l`, o comando `jobs` exibe adicionalmente o identificador do processo (PID) logo antes do status:

```
$ jobs -l
[1]+ 1114 Stopped                 sleep 60
```

As demais opções possíveis para jobs são:

-n

Lista apenas os processos que mudaram de status desde a última notificação. Os possíveis status incluem Running, Stopped, Terminated ou Done.

-p

Lista os IDs do processo.

-r

Lista apenas os jobs em execução.

-s

Lista apenas os jobs interrompidos (ou suspensos).

NOTE

Lembre-se, um job tem um *ID de trabalho* e um *ID de processo* (PID).

Especificação do trabalho

O comando `jobs`, a exemplo de outros utilitários como `fg`, `bg` e `kill` (que você verá na próxima seção), precisa de uma especificação de trabalho (ou `jobspec`) para agir sobre um job particular. Como acabamos de ver, este pode ser – e normalmente é – o ID do trabalho precedido por %. No entanto, outras especificações de trabalho também são possíveis. Vamos dar uma olhada nelas:

%n

Job cujo número de ID é n:

```
$ jobs %1  
[1]+  Stopped                  sleep 60
```

%str

Job cuja linha de comando começa com str:

```
$ jobs %sl  
[1]+  Stopped                  sleep 60
```

?str

Job cuja linha de comando contém str:

```
$ jobs %?le
```

```
[1]+ Stopped                 sleep 60
```

%+ ou %%

Job atual (o que foi iniciado por último em segundo plano ou suspenso do primeiro plano):

```
$ jobs %+  
[1]+ Stopped                 sleep 60
```

%-

Job anterior (aquele que era %+ antes do padrão, que é o atual):

```
$ jobs %-  
[1]+ Stopped                 sleep 60
```

No nosso caso, como há apenas um job, ele é o atual e o anterior.

Status do trabalho: suspensão, primeiro plano e segundo plano

Uma vez que um job está em segundo plano ou foi suspenso, podemos fazer três coisas com ele:

1. Levá-lo ao primeiro plano com fg:

```
$ fg %1  
sleep 60
```

fg move o job especificado para o primeiro plano e o torna o job atual. A seguir podemos esperar que ele seja concluído, interrompê-lo novamente com Ctrl + Z ou encerrá-lo com Ctrl + C.

2. Colocá-lo em segundo plano com bg:

```
$ bg %1  
[1]+ sleep 60 &
```

Uma vez no segundo plano, o trabalho pode ser trazido de volta ao primeiro plano com fg ou eliminado (veja abaixo). Observe o e comercial (&) indicando que o trabalho foi enviado para o segundo plano. Na verdade, também podemos usar o e comercial para iniciar um processo diretamente em segundo plano:

```
$ sleep 100 &
[2] 970
```

Junto com o ID do novo trabalho ([2]), também obtemos o ID do processo (970). Agora, ambos os trabalhos estão rodando em segundo plano:

```
$ jobs
[1]-  Running                 sleep 60 &
[2]+  Running                 sleep 100 &
```

Um pouco mais tarde, o primeiro trabalho termina de ser executado:

```
$ jobs
[1]-  Done                   sleep 60
[2]+  Running                 sleep 100 &
```

3. Encerrá-lo através de um sinal SIGTERM com `kill`:

```
$ kill %2
```

Para ter certeza de que o job foi encerrado, rode `jobs` novamente:

```
$ jobs
[2]+  Terminated             sleep 100
```

NOTE

Se nenhum trabalho for especificado, `fg` e `bg` agirão sobre o job padrão atual. `kill`, no entanto, sempre precisa de uma especificação de trabalho.

Trabalhos desvinculados: `nohup`

Os jobs que vimos nas seções anteriores estavam todos vinculados à sessão do usuário que os invocou. Isso significa que, se a sessão for encerrada, os jobs serão perdidos. No entanto, é possível desvincular jobs de sessões e executá-los mesmo após o encerramento da sessão. Isso é feito com o comando `nohup` ("no hangup"). A sintaxe é a seguinte:

```
nohup COMMAND &
```

Lembre-se, o & envia o processo para o segundo plano e libera o terminal em que estamos trabalhando.

Vamos desvincular o job em segundo plano ping localhost da sessão atual:

```
$ nohup ping localhost &
[1] 1251
$ nohup: ignoring input and appending output to 'nohup.out'
^C
```

A saída mostra o ID do trabalho ([1]) e o PID (1251), seguido por uma mensagem nos informando sobre o arquivo nohup.out. Este é o arquivo padrão no qual stdout e stderr serão salvos. Agora podemos pressionar Ctrl + C para liberar o prompt de comando, fechar a sessão, iniciar outra como root e usar tail -f para verificar se o comando está rodando e a saída está sendo escrita no arquivo padrão:

```
$ exit
logout
$ tail -f /home/carol/nohup.out
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from localhost (::1): icmp_seq=5 ttl=64 time=0.070 ms
^C
```

TIP

Em vez de usar o nohup.out padrão, poderíamos ter especificado um arquivo de saída à escolha com nohup ping localhost > /path/to/your/file &.

Se quisermos encerrar o processo, devemos especificar seu PID:

```
# kill 1251
```

Monitoramento de processos

Um processo ou tarefa é uma instância de um programa em execução. Assim, criamos novos processos toda vez que digitamos comandos no terminal.

O comando watch executa um programa periodicamente (por padrão, a cada 2 segundos) e nos permite *observar* a mudança da saída do programa ao longo do tempo. Por exemplo, podemos monitorar como a média de trabalho muda conforme mais processos são executados digitando watch

`uptime`:

```
Every 2.0s: uptime          debian: Tue Aug 20 23:31:27 2019
23:31:27 up 21 min,  1 user,  load average: 0.00, 0.00, 0.00
```

O comando roda até ser interrompido, então teríamos de pará-lo com `Ctrl + C`. Obtemos duas linhas na saída: a primeira corresponde ao `watch` e nos informa a frequência com que o comando será executado (`Every 2.0s: uptime`), qual o comando/programa a observar (`uptime`) além do nome do host e a data (`debian: Tue Aug 20 23:31:27 2019`). A segunda linha da saída é o tempo de atividade e inclui a hora (`23:31:27`), o tempo em que o sistema está ativo (`up 21 min`), o número de usuários ativos (`1 user`) e a carga média do sistema ou o número de processos em execução ou em estado de espera nos últimos 1, 5 e 15 minutos (`load average: 0.00, 0.00, 0.00`).

Da mesma forma, você pode verificar o uso de memória à medida que novos processos são criados com `watch free`:

```
Every 2.0s: free          debian: Tue Aug 20 23:43:37 2019
23:43:37 up 24 min,  1 user,  load average: 0.00, 0.00, 0.00
              total        used        free      shared  buff/cache   available
Mem:       16274868       493984     14729396       35064     1051488      15462040
Swap:      16777212           0     16777212
```

Para alterar o intervalo de atualização de `watch`, use as opções `-n` ou `--interval`, mais o número de segundos, como em:

```
$ watch -n 5 free
```

Agora o comando `free` será executado a cada 5 segundos.

Para saber mais sobre as opções de `uptime`, `free` e `watch`, consulte as páginas de manual correspondentes.

NOTE

As informações fornecidas por `uptime` e `free` também são integradas nas ferramentas mais abrangentes `top` e `ps` (veja abaixo).

Enviando sinais para processos: kill

Cada processo possui um identificador de processo ou PID exclusivo. Uma maneira de descobrir o PID de um processo é usar o comando `pgrep` seguido pelo nome do processo:

```
$ pgrep sleep
1201
```

NOTE

O identificador de um processo também pode ser descoberto com o comando `pidof` (p.ex. `pidof sleep`).

Como no caso do `pgrep`, o comando `pkill` elimina um processo com base em seu nome:

```
$ pkill sleep
[1]+ Terminated                 sleep 60
```

Para eliminar várias instâncias do mesmo processo, o comando `killall` pode ser usado:

```
$ sleep 60 &
[1] 1246
$ sleep 70 &
[2] 1247
$ killall sleep
[1]- Terminated                 sleep 60
[2]+ Terminated                 sleep 70
```

Tanto `pkill` quanto `killall` funcionam da mesma maneira que `kill`, ou seja, enviam um sinal de encerramento para o(s) processo(s) especificado(s). Se nenhum sinal for fornecido, o padrão SIGTERM é enviado. No entanto, `kill` só aceita um ID de trabalho ou de processo como argumento.

Os sinais podem ser especificados por:

- Nome:

```
$ kill -SIGHUP 1247
```

- Número:

```
$ kill -1 1247
```

- Opção:

```
$ kill -s SIGHUP 1247
```

Para fazer com que `kill` funcione de forma semelhante a `pkill` ou `killall` (evitando os comandos para descobrir os PIDs correspondentes), podemos usar a substituição de comandos:

```
$ kill -1 $(pgrep sleep)
```

Como você já deve saber, uma sintaxe alternativa é `kill -1 'pgrep sleep'`.

TIP

Para uma lista exaustiva de todos os sinais de `kill` e seus códigos, digite `kill -l` no terminal. Use `-KILL` (`-9` ou `-s KILL`) para eliminar processos rebeldes quando todos os outros sinais falharem.

top e ps

Quando se trata de monitoramento de processos, duas ferramentas inestimáveis são `top` e `ps`. Enquanto o primeiro produz resultados de maneira dinâmica, o último o faz de maneira estática. Em todos os casos, ambos são excelentes utilitários para se ter uma visão abrangente de todos os processos do sistema.

Interação com top

Para chamar o `top`, basta digitar `top`:

```
$ top
```

```
top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14
Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache
KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem

PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
 436 carol    20   0    42696    3624    3060 R  0,7  0,4  0:00.30 top
    4 root     20   0        0        0        0 S  0,3  0,0  0:00.12 kworker/0:0
```

399	root	20	0	95204	6748	5780	S	0,3	0,7	0:00.22	sshd
1	root	20	0	56872	6596	5208	S	0,0	0,6	0:01.29	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.02	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/u2:0
7	root	20	0	0	0	0	S	0,0	0,0	0:00.08	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
10	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	lru-add-drain
(...)											

O `top` permite uma certa interação do usuário. Por padrão, a saída é classificada pela porcentagem de tempo da CPU usada por cada processo em ordem decrescente. Esse comportamento pode ser modificado pressionando as seguintes teclas de dentro do `top`:

M

Classificar por uso da *memória*.

N

Classificar pelo *número ID* do processo.

T

Classificar por *tempo* de execução.

P

Classificar por *porcentagem* de uso da CPU.

TIP Para alternar entre a ordem crescente/decrescente, basta pressionar R.

Outras teclas interessantes para interagir com `top` são:

? ou h

Ajuda.

k

Elimina um processo. O `top` pedirá o PID do processo a encerrar, assim como o sinal a ser enviado (por padrão, SIGTERM ou 15).

r

Altera a prioridade de um processo (`renice`). `top` pede o valor de nice. Os valores possíveis

variam de -20 a 19, mas apenas o superusuário (`root`) pode definir um valor negativo ou inferior ao atual.

u

Lista os processos de um determinado usuário (por padrão, são mostrados os processos de todos os usuários).

c

Mostra os caminhos absolutos dos programas e diferencia os processos do espaço do usuário dos processos do espaço do kernel (entre colchetes).

v

Visão de floresta/hierárquica dos processos.

t e m

Mudam a aparência das leituras da CPU e da memória, respectivamente, em um ciclo de quatro estágios: os dois primeiros pressionamentos mostram barras de progresso, o terceiro oculta a barra e o quarto a traz de volta.

w

Salva as definições de configuração em `~/.toprc`.

TIP

Uma versão mais sofisticada e amigável de `top` é `htop`. Outra alternativa, talvez mais exaustiva, é `atop`. Se ainda não estiverem instalados em seu sistema, use seu gerenciador de pacotes para instalá-los e experimentá-los.

Explicação da saída de `top`

A saída de `top` é dividida em duas áreas: a *área de resumo* e a *área de tarefas*.

A área de resumo em `top`

A *área de resumo* é composta pelas cinco linhas superiores e nos fornece as seguintes informações:

- `top` – 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14
 - hora atual (em formato de 24 horas): 11:20:29
 - tempo de atividade (há quanto tempo o sistema está ativo e funcionando): up 2:21
 - número de usuários logados e carga média da CPU nos últimos 1, 5 e 15 minutos, respectivamente: load average: 0,11, 0,20, 0,14

- Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie (informações sobre os processos)
 - número total de processos em modo ativo: 73 total
 - em execução (os que estão sendo executados): 1 running
 - em espera (os que estão esperando para retomar a execução): 72 sleeping
 - interrompidos (por um sinal de controle do trabalho): 0 stopped
 - zumbi (os que concluíram a execução mas ainda estão esperando que o processo pai os remova da tabela de processos): 0 zombie
- %Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st (porcentagem de tempo da CPU gasto em:)
 - processos de usuário: 0,0 us
 - processos do sistema/kernel: 0,4 sy
 - processos com um valor *nice* configurado—quanto mais alto o valor nice, menor a prioridade: 0,0 ni
 - nada—tempo ocioso da CPU: 99,7 id
 - processos aguardando operações de I/O: 0,0 wa
 - processos atendendo interrupções de hardware—periféricos enviando ao processador sinais que precisam de atenção: 0,0 hi
 - processos atendendo interrupções de software: 0,0 si
 - processos atendendo tarefas de outras máquinas virtuais em um ambiente virtual, e que portanto roubam tempo: 0,0 st
- KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache (informações da memória em kilobytes)
 - quantidade total de memória: 1020332 total
 - memória não utilizada: 909492 free
 - memória em uso: 38796 used
 - memória armazenada em buffer e em cache para evitar acesso excessivo ao disco: 72044 buff/cache

Note como o total é a soma dos outros três valores—free, used e buff/cache—(aproximadamente 1 GB em nosso caso).

- KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem (informações de troca em kilobytes)
 - quantidade total de espaço de troca: 1046524 total
 - espaço de troca não utilizado: 1046524 free
 - espaço de troca em uso: 0 used
 - quantidade de memória de troca que pode ser alocada a processos sem causar mais trocas: 873264 avail Mem

A área de tarefas em `top`: Campos e colunas

Abaixo da *área de resumo* fica a *área de tarefas*, que inclui uma série de *campos* e *colunas* que informam sobre os processos em execução:

PID

Identificador do processo.

USER

Usuário emissor do comando que originou o processo

PR

Prioridade de processo para o kernel.

NI

Valor nice do processo. Os valores mais baixos têm mais prioridade que os valores altos.

VIRT

Quantidade total de memória usada por processo (incluindo Troca).

RES

Memória RAM usada por processo.

SHR

Memória compartilhada do processo com outros processos.

S

Status do processo. Os valores incluem: S (suspenção interrompível—esperando que um evento termine), R (executável—em execução ou na fila para ser executado) ou Z (zumbi—processos filhos encerrados cujas estruturas de dados ainda não foram removidas da tabela de processos).

%CPU

Porcentagem de CPU usada pelo processo.

%MEM

Porcentagem de RAM utilizada pelo processo, ou seja, o valor de RES expresso em porcentagem.

TIME+

Tempo total de atividade do processo.

COMMAND

Nome do comando/programa que gerou o processo.

Visualizando processos estaticamente: ps

Como dito acima, o `ps` exibe um instantâneo dos processos. Para ver todos os processos com um terminal (tty), digite `ps a`:

```
$ ps a
 PID TTY      STAT    TIME COMMAND
 386 tty1    Ss+   0:00 /sbin/agetty --noclear tty1 linux
 424 tty7    Ssl+  0:00 /usr/lib/xorg/Xorg :0 -seat seat0 (...)
 655 pts/0    Ss    0:00 -bash
1186 pts/0    R+    0:00 ps a
(...)
```

Explicação da sintaxe e da saída das opções de ps

Com relação às opções, o `ps` aceita três estilos diferentes: BSD, UNIX e GNU. Vamos ver como cada um desses estilos se comportaria ao relatar informações sobre a ID de um processo específico:

BSD

As opções não requerem um traço inicial:

```
$ ps p 811
 PID TTY      STAT    TIME COMMAND
 811 pts/0    S      0:00 -su
```

UNIX

As opções requerem um traço inicial:

```
$ ps -p 811
PID TTY          TIME CMD
811 pts/0        00:00:00 bash
```

GNU

As opções recebem um duplo traço inicial:

```
$ ps --pid 811
PID TTY          TIME CMD
811 pts/0        00:00:00 bash
```

Nos três casos, o `ps` relata informações sobre o processo cujo PID é 811—neste caso, o `bash`.

Da mesma forma, podemos usar o `ps` para pesquisar os processos iniciados por um usuário determinado:

- `ps U carol` (BSD)
- `ps -u carol` (UNIX)
- `ps --user carol` (GNU)

Vamos verificar os processos iniciados por `carol`:

```
$ ps U carol
PID TTY      STAT   TIME COMMAND
811 pts/0    S      0:00 -su
898 pts/0    R+     0:00 ps U carol
```

Ela iniciou dois processos: `bash (-su)` e `ps (ps U carol)`. A coluna `STAT` informa o estado do processo (veja abaixo).

Podemos obter o melhor do `ps` combinando algumas de suas opções. Um comando muito útil (produzindo uma saída semelhante à de `top`) é `ps aux` (estilo BSD). Nesse caso, os processos de todos os shells (não apenas o atual) são mostrados. O significado das opções é o seguinte:

a

Mostra processos que estão vinculados a um `tty` ou terminal.

u

Exibe formato orientado ao usuário.

x

Mostra processos que não estão vinculados a um tty ou terminal.

```
$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      1  0.0  0.1 204504  6780 ?        Ss 14:04  0:00 /sbin/init
root      2  0.0  0.0     0     0 ?        S  14:04  0:00 [kthreadd]
root      3  0.0  0.0     0     0 ?        S  14:04  0:00 [ksoftirqd/0]
root      5  0.0  0.0     0     0 ?        S< 14:04  0:00 [kworker/0:0H]
root      7  0.0  0.0     0     0 ?        S  14:04  0:00 [rcu_sched]
root      8  0.0  0.0     0     0 ?        S  14:04  0:00 [rcu_bh]
root      9  0.0  0.0     0     0 ?        S  14:04  0:00 [migration/0]
(...)
```

Entenda melhor as colunas:

USER

Proprietário do processo.

PID

Identificador do processo.

%CPU

Porcentagem de CPU usada.

%MEM

Porcentagem de memória física usada.

VSZ

Memória virtual do processo em KiB.

RSS

Memória física não trocada usada pelo processo em KiB.

TT

Terminal (tty) que controla o processo.

STAT

Código que representa o estado do processo. Além de S, R e Z (que vimos ao descrever a saída de `top`), outros valores possíveis seriam: D (dormente sem interrupção – geralmente esperando por I/O), T (interrompido – normalmente por um sinal de controle). Alguns modificadores extras incluem: < (prioridade maior que o convencional), N (prioridade menor que o convencional), ou + (no grupo de processos em primeiro plano).

STARTED

Horário de início do processo.

TIME

Tempo de CPU acumulado.

COMMAND

Comando que iniciou o processo.

Exercícios Guiados

1. `oneko` é um programa divertido que mostra um gato perseguindo o cursor do mouse. Se ele ainda não estiver instalado em seu sistema desktop, instale-o usando o gerenciador de pacotes de sua distribuição. Vamos usá-lo para estudar o controle de jobs.
 - Inicie o programa. Qual o procedimento?

- Mova o cursor do mouse para ver como o gato o persegue. Agora suspenda o processo. Qual o procedimento? Qual é a saída?

- Verifique quantos jobs estão presentes atualmente. O que você digita? Qual é a saída?

- Agora envie-o para o segundo plano especificando seu ID de trabalho. Qual é a saída? Como você pode saber se o job está sendo executado em segundo plano?

- Finalmente, encerre o job especificando seu ID de trabalho. O que você digita?

2. Descubra o PID de todos os processos gerados pelo servidor web *Apache HTTPD* (`apache2`) com dois comandos diferentes:

3. Encerre todos os processos `apache2` sem usar seus PIDs e com dois comandos diferentes:

4. Suponha que você tem de encerrar todas as instâncias do `apache2` e não tem tempo para descobrir quais são os PIDs correspondentes. Como fazer isso usando `kill` com o sinal padrão `SIGTERM` em uma só linha?

5. Inicie o `top` e interaja com ele executando o seguinte:

- Exibir uma visão em floresta dos processos:

- Exibir caminhos completos de processos, diferenciando entre espaço de usuário e espaço de kernel:

6. Digite o comando `ps` para exibir todos os processos iniciados pelo usuário do servidor web *Apache HTTPD* (`www-data`):

- Usando a sintaxe do BSD:

- Usando a sintaxe do UNIX:

- Usando a sintaxe do GNU:

Exercícios Exploratórios

1. O sinal `SIGHUP` pode ser usado como forma de reiniciar certos daemons. Com o servidor web *Apache HTTPD*—por exemplo—enviar `SIGHUP` para o processo pai (iniciado por `init`) elimina seus filhos. O pai, no entanto, relê seus arquivos de configuração, reabre os arquivos de log e gera um novo conjunto de filhos. Execute as seguintes tarefas:

- Inicie o servidor web:

- Certifique-se de conhecer o PID do processo pai:

- Faça o servidor web Apache HTTPD reiniciar enviando o sinal `SIGHUP` para o processo pai:

- Verifique se o pai não foi eliminado e se novos filhos foram gerados:

2. Embora inicialmente estática, a saída de `ps` pode ser *tornada* dinâmica combinando `ps` e `watch`. Vamos monitorar as novas conexões do servidor web *Apache HTTPD*. Antes de realizar as tarefas descritas abaixo, recomendamos ler a descrição da diretiva `MaxConnectionsPerChild` em [Apache MPM Common Directives](#).

- Adicione a diretiva `MaxConnectionsPerChild` com um valor de `1` no arquivo de configuração de `apache2`—no *Debian* e seus derivativos ele fica em `/etc/apache2/apache2.conf`; na família *CentOS*, em `/etc/httpd/conf/httpd.conf`. Não esqueça de reiniciar `apache2` para que as mudanças sejam aplicadas.

- Digite um comando que use `watch`, `ps` e `grep` para as conexões do `apache2`.

- Agora abra um navegador web ou use um navegador de linha de comando como o `lynx` para estabelecer uma conexão com o servidor web através de seu endereço IP. O que você observa na saída de `watch`?

3. Como vimos, por padrão, `top` classifica as tarefas por porcentagem de uso da CPU em ordem decrescente (com os valores mais altos no topo). Esse comportamento pode ser modificado com as teclas interativas `M` (uso de memória), `N` (identificador único do processo), `T` (tempo de execução) e `P` (porcentagem de tempo de CPU). No entanto, também podemos classificar a lista de tarefas a gosto, iniciando `top` com a opção `-o` (para saber mais, verifique a página `man` de `top`). Agora, execute as seguintes tarefas:

- Inicie o `top` para que as tarefas sejam classificadas por uso da memória:

- Verifique se digitou o comando correto destacando a coluna da memória:

4. O `ps` também tem uma opção `-o` para especificar as colunas que você deseja mostrar. Estude esta opção e execute as seguintes tarefas:

- Inicie o `ps` para exibir somente informações sobre o *usuário*, *porcentagem de memória usada*, *porcentagem de tempo da CPU usado* e *comando completo*:

- Agora, inicie o `ps` para que as únicas informações exibida sejam o usuário e o nome dos programas que ele está usando:

Resumo

Nesta lição, você aprendeu sobre *jobs* (trabalhos) e *controle de jobs*. Os fatos e conceitos principais a serem lembrados são:

- Jobs são processos enviados ao segundo plano.
- Além de um *ID de processo*, os jobs também recebem um *ID de trabalho* quando criados.
- Para controlar os jobs, é necessária uma especificação de trabalho (*jobspec*).
- Os jobs podem ser trazidos ao primeiro plano, enviados ao segundo plano, encerrados e eliminados (ou *mortos*).
- Um job pode ser desvinculado do terminal e da sessão na qual foi criado.

Discutimos também o conceito de *processos* e *monitoramento de processos*. As ideias mais relevantes são:

- Os processos são programas em execução.
- Os processos podem ser monitorados.
- Diferentes utilitários nos permitem descobrir o *ID do processo* dos processos, bem como enviar sinais para encerrá-los.
- Os sinais podem ser especificados por nome (p. ex. `-SIGTERM`), número (p. ex. `-15`) ou opção (p. ex. `-s SIGTERM`).
- `top` e `ps` são muito poderosos quando se trata de monitorar processos. A saída do `top` é dinâmica e se atualiza constantemente; já o `ps` exibe a saída de forma estática.

Comandos usados nesta lição:

jobs

Mostra os jobs ativos e seus status.

sleep

Espera por um período específico de tempo.

fg

Traz o job para o primeiro plano.

bg

Move o job para o segundo plano.

kill

Elimina o job.

nohup

Desvincula o job da sessão/terminal.

exit

Sai do shell atual.

tail

Exibe as linhas mais recentes em um arquivo.

watch

Executa um comando repetidamente (ciclo de 2 segundos por padrão).

uptime

Mostra há quanto tempo o sistema está rodando, o número de usuários atuais e a carga média do sistema.

free

Mostra o uso da memória.

pgrep

Procura o ID do processo com base no nome.

pidof

Procura o ID do processo com base no nome.

pkill

Envia sinal ao processo por nome.

killall

Elimina processo(s) por nome.

top

Exibe os processos do Linux.

ps

Relata um instantâneo dos processos atuais.

Respostas aos Exercícios Guiados

1. `oneko` é um programa divertido que mostra um gato perseguindo o cursor do mouse. Se ele ainda não estiver instalado em seu sistema desktop, instale-o usando o gerenciador de pacotes de sua distribuição. Vamos usá-lo para estudar o controle de jobs.

- Inicie o programa. Qual o procedimento?

Digitar `oneko` no terminal.

- Mova o cursor do mouse para ver como o gato o persegue. Agora suspenda o processo. Qual o procedimento? Qual é a saída?

Pressionar a combinação de teclas `Ctrl + z`:

```
[1]+ Stopped      oneko
```

- Verifique quantos jobs estão presentes atualmente. O que você digita? Qual é a saída?

```
$ jobs
[1]+ Stopped      oneko
```

- Agora envie-o para o segundo plano especificando seu ID de trabalho. Qual é a saída? Como você pode saber se o job está sendo executado em segundo plano?

```
$ bg %1
[1]+ oneko &
```

O gato está se movendo outra vez.

- Finalmente, encerre o job especificando seu ID de trabalho. O que você digita?

```
$ kill %1
```

2. Descubra o PID de todos os processos gerados pelo servidor web *Apache HTTPD* (`apache2`) com dois comandos diferentes:

```
$ pgrep apache2
```

ou

```
$ pidof apache2
```

3. Encerre todos os processos apache2 sem usar seus PIDs e com dois comandos diferentes:

```
$ pkill apache2
```

ou

```
$ killall apache2
```

4. Suponha que você tem de encerrar todas as instâncias do apache2 e não tem tempo para descobrir quais são os PIDs correspondentes. Como fazer isso usando kill com o sinal padrão SIGTERM em uma só linha?

```
$ kill $(pgrep apache2)  
$ kill `pgrep apache2`
```

ou

```
$ kill $(pidof apache2)  
$ kill `pidof apache2`
```

NOTE

Como SIGTERM (15) é o sinal padrão, não é necessário passar nenhuma opção para kill.

5. Inicie o top e interaja com ele executando o seguinte:

- Exibir uma visão em floresta dos processos:

Pressione V.

- Exibir caminhos completos de processos, diferenciando entre espaço de usuário e espaço de kernel:

Pressione C.

6. Digite o comando `ps` para exibir todos os processos iniciados pelo usuário do servidor web *Apache HTTPD* (`www-data`):

- Usando a sintaxe do BSD:

```
$ ps U www-data
```

- Usando a sintaxe do UNIX:

```
$ ps -u www-data
```

- Usando a sintaxe do GNU:

```
$ ps --user www-data
```

Respostas aos Exercícios Exploratórios

1. O sinal SIGHUP pode ser usado como forma de reiniciar certos daemons. Com o servidor web *Apache HTTPD*—por exemplo—enviar SIGHUP para o processo pai (iniciado por `init`) elimina seus filhos. O pai, no entanto, relê seus arquivos de configuração, reabre os arquivos de log e gera um novo conjunto de filhos. Execute as seguintes tarefas:

- Inicie o servidor web:

```
$ sudo systemctl start apache2
```

- Certifique-se de conhecer o PID do processo pai:

```
$ ps aux | grep apache2
```

O processo pai é aquele iniciado pelo usuário `root`. Em nosso caso, o PID é 1653.

- Faça o servidor web Apache HTTPD reiniciar enviando o sinal SIGHUP para o processo pai:

```
$ kill -SIGHUP 1653
```

- Verifique se o pai não foi eliminado e se novos filhos foram gerados:

```
$ ps aux | grep apache2
```

Agora você deve ver o processo pai `apache2` junto com dois novos filhos.

2. Embora inicialmente estática, a saída de `ps` pode ser *tornada* dinâmica combinando `ps` e `watch`. Vamos monitorar as novas conexões do servidor web *Apache HTTPD*. Antes de realizar as tarefas descritas abaixo, recomendamos ler a descrição da diretiva `MaxConnectionsPerChild` em [Apache MPM Common Directives](#).

- Adicione a diretiva `MaxConnectionsPerChild` com um valor de 1 no arquivo de configuração de `apache2`—no *Debian* e seus derivativos ele fica em `/etc/apache2/apache2.conf`; na família *CentOS*, em `/etc/httpd/conf/httpd.conf`. Não esqueça de reiniciar `apache2` para que as mudanças sejam aplicadas.

A linha a incluir no arquivo de configuração é `MaxConnectionsPerChild 1`. Uma maneira de reiniciar o servidor web é através de `sudo systemctl restart apache2`.

- Digite um comando que use `watch`, `ps` e `grep` para as conexões do apache2.

```
$ watch 'ps aux | grep apache2'
```

ou

```
$ watch "ps aux | grep apache2"
```

- Agora abra um navegador web ou use um navegador de linha de comando como o `lynx` para estabelecer uma conexão com o servidor web através de seu endereço IP. O que você observa na saída de `watch`?

Um dos processos filho de propriedade de `www-data` desaparece.

- Como vimos, por padrão, `top` classifica as tarefas por porcentagem de uso da CPU em ordem decrescente (com os valores mais altos no topo). Esse comportamento pode ser modificado com as teclas interativas `M` (uso de memória), `N` (identificador único do processo), `T` (tempo de execução) e `P` (porcentagem de tempo de CPU). No entanto, também podemos classificar a lista de tarefas a gosto, iniciando `top` com a opção `-o` (para saber mais, verifique a página `man` de `top`). Agora, execute as seguintes tarefas:

- Inicie o `top` para que as tarefas sejam classificadas por uso da memória:

```
$ top -o %MEM
```

- Verifique se digitou o comando correto destacando a coluna da memória:

Pressione `x`.

- O `ps` também tem uma opção `o` para especificar as colunas que você deseja mostrar. Estude esta opção e execute as seguintes tarefas:

- Inicie o `ps` para exibir somente informações sobre o *usuário, porcentagem de memória usada, porcentagem de tempo da CPU usado e comando completo*:

```
$ ps o user,%mem,%cpu,cmd
```

- Agora, inicie o `ps` para que as únicas informações exibida sejam o usuário e o nome dos programas que ele está usando:

```
$ ps o user,comm
```



103.5 Lição 2

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.5 Criar, monitorar e eliminar processos
Lição:	2 de 2

Introdução

As ferramentas e utilitários vistos na lição anterior são muito úteis para o monitoramento de processos em geral. No entanto, o administrador do sistema pode precisar ir além. Nesta lição, discutiremos o conceito de multiplexadores de terminal e aprenderemos sobre *GNU Screen* e *tmux*, já que—embora os emuladores de terminal modernos sejam excelentes—os multiplexadores ainda preservam alguns recursos interessantes e poderosos para a produtividade de um administrador de sistema.

Recursos dos multiplexadores de terminal

Em eletrônica, um multiplexador (ou *mux*) é um dispositivo que permite que várias entradas sejam conectadas a uma única saída. Assim, um multiplexador de terminal nos proporciona a capacidade de alternar entre diferentes entradas conforme necessário. Embora não sejam exatamente iguais, *screen* e *tmux* compartilham uma série de características comuns:

- Qualquer invocação bem-sucedida resultará em pelo menos uma sessão que—por sua vez—incluirá ao menos uma janela. As janelas contêm programas.
- As janelas podem ser divididas em regiões ou painéis—o que ajuda na produtividade ao se

trabalhar com vários programas simultaneamente.

- Facilidade de controle: para executar a maioria dos comandos, usamos uma combinação de teclas – o chamado *prefixo de comando* ou *chave de comando* – seguida por outro caractere.
- As sessões podem ser desanexadas do terminal em que estão (ou seja, os programas são enviados para o segundo plano e continuam em execução). Isso garante a execução completa dos programas, independentemente de fecharmos accidentalmente um terminal, travamentos ocasionais ou até mesmo a perda da conexão remota.
- Conexão de socket.
- Modo de cópia.
- São altamente personalizáveis.

GNU Screen

Nos primórdios do Unix (anos 1970-80), os computadores consistiam basicamente em terminais conectados a um computador central. Era só isso mesmo, sem um monte de janelas ou abas. E essa foi a razão por trás da criação do GNU Screen em 1987: emular múltiplas telas VT100 independentes em um único terminal físico.

Janelas

O GNU Screen é invocado simplesmente digitando `screen` no terminal. Aparece primeiro uma mensagem de boas-vindas:

```
GNU Screen version 4.05.00 (GNU) 10-Dec-16
```

```
Copyright (c) 2010 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008, 2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul
Habib Chowdhury
Copyright (c) 1993-2002, 2003, 2005, 2006, 2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann
(...)
```

Pressione Espaço ou Enter para fechar a mensagem e será mostrado um prompt de comando:

```
$
```

Pode parecer que nada aconteceu, mas o fato é que o comando `screen` já criou e gerencia sua primeira sessão e janela. O prefixo do comando `screen` é `Ctrl + a`. Para ver todas as janelas na parte

inferior da tela do terminal, digite **Ctrl + a - w**:

```
0*$ bash
```

Aqui está, nossa única janela até agora! Observe, no entanto, que a contagem começa em 0. Para criar outra janela, digite **Ctrl + a - c**. Você verá um novo prompt. Vamos iniciar o **ps** nessa nova janela:

```
$ ps
PID TTY          TIME CMD
974 pts/2    00:00:00 bash
981 pts/2    00:00:00 ps
```

e digitar **Ctrl + a - w** novamente:

```
0-$ bash 1*$ bash
```

Agora temos nossas duas janelas (observe o asterisco que indica a que está sendo exibida no momento). No entanto, como foram iniciadas com o Bash, as duas receberam o mesmo nome. Já que chamamos o **ps** em nossa janela atual, vamos renomeá-lo com esse mesmo nome. Para isso, digite **Ctrl + a - A** e escreva o novo nome da janela (**ps**) quando solicitado:

```
Set window's title to: ps
```

Agora, vamos criar outra janela, já com um nome desde o início: **yetanotherwindow**. Para isso, invocamos **screen** com a opção **-t**:

```
$ screen -t yetanotherwindow
```

Há várias maneiras de se mover entre as janelas:

- Usando **Ctrl + a - n** (ir para a próxima janela, *next* em inglês) e **Ctrl + a - p** (ir para a janela anterior, *previous*).
- Usando **Ctrl + a - número** (ir para a janela número *número*).
- Usando **Ctrl + a - "** para ver uma lista de todas as janelas. Use as setinhas do teclado para selecionar a janela que deseja e dê Enter:

Num	Name	Flags
0	bash	\$
1	ps	\$
2	yetanotherwindow	

Ao trabalhar com janelas, é importante lembrar o seguinte:

- As janelas executam seus programas de forma totalmente independente umas das outras.
- Os programas continuarão a ser executados mesmo que suas janelas não estejam visíveis (inclusive quando a sessão de screen for desanexada, como veremos em breve).

Para remover uma janela, basta encerrar o programa que está em execução nela (quando a última janela for removida, o próprio screen será encerrado). Outra alternativa é usar **Ctrl + a - k** enquanto estiver na janela que deseja remover; será solicitada a confirmação:

Really kill this window [y/n]

Window 0 (bash) killed.

Regiões

O screen pode dividir uma tela de terminal em diversas regiões para acomodar janelas. Essas divisões podem ser horizontais (**Ctrl + a - S**) ou verticais (**Ctrl + a - l**).

A única coisa que a nova região mostrará é **--** na parte inferior, o que significa que está vazia:

1 ps

--

Para passar à nova região, digite **Ctrl + a - Tab**. Agora podemos adicionar uma janela com qualquer um dos métodos que já vimos, por exemplo: **Ctrl + a - 2**. Assim, o **--** deve se transformar em 2 yetanotherwindow:

```
$ ps
PID TTY          TIME CMD
1020 pts/2    00:00:00 bash
1033 pts/2    00:00:00 ps
$ screen -t yetanotherwindow
```

1 ps
yetanotherwindow

2

Alguns aspectos importantes a lembrar ao se trabalhar com regiões:

- Para se mover entre regiões, digite `Ctrl l + a - Tab`.
- Para encerrar todas as regiões exceto a atual, use `Ctrl l + a - Q`.
- Para encerrar a região atual, use `Ctrl l + a - X`.
- O encerramento de uma região não encerra a janela associada a ela.

Sessões

Até agora brincamos com algumas janelas e regiões, mas todas pertencentes à mesma e única sessão. É hora de começar a brincar com as sessões. Para ver uma lista de todas as sessões, digite `screen -list` ou `screen -ls`:

```
$ screen -list
There is a screen on:
    1037.pts-0.debian          (08/24/19 13:53:35)      (Attached)
1 Socket in /run/screen/S-carol.
```

Essa é a nossa única sessão no momento:

PID

1037

Nome

`pts-0.debian` (indicando o terminal – em nosso caso, um *pseudo-terminal escravo* – e o nome do host).

Status

Attached

Vamos criar uma nova sessão, dando a ela um nome mais descriptivo:

```
$ screen -S "second session"
```

A tela do terminal será esvaziada e um novo prompt se abrirá. Você pode verificar as sessões mais uma vez:

```
$ screen -ls
There are screens on:
    1090.second session      (08/24/19 14:38:35)      (Attached)
    1037.pts-0.debian        (08/24/19 13:53:36)      (Attached)
2 Sockets in /run/screen/S-carol.
```

Para encerrar uma sessão, saia de todas as janelas ou simplesmente digite o comando `screen -S SESSION-PID -X quit` (também é possível fornecer o nome da sessão). Vamos nos livrar de nossa primeira sessão:

```
$ screen -S 1037 -X quit
```

Você será enviado de volta ao prompt do terminal fora de `screen`. Mas lembre-se, nossa segunda sessão ainda está viva:

```
$ screen -ls
There is a screen on:
    1090.second session (08/24/19 14:38:35) (Detached)
1 Socket in /run/screen/S-carol.
```

No entanto, como eliminamos a sessão pai, ela recebe um novo rótulo: `Detached` (desanexada).

Desanexando sessões

Por uma série de razões, podemos querer desanexar uma sessão do `screen` do terminal a que pertence:

- Para deixar o computador da empresa cumprir seu dever e conectar-se remotamente mais tarde, de casa.
- Para compartilhar uma sessão com outros usuários.

Para desanexar uma sessão, a combinação de teclas é `Ctrl + a -d`. Você é levado de volta ao seu terminal:

```
[detached from 1090.second session]
$
```

Para entrar novamente na sessão desanexada, use o comando `screen -r SESSION-PID`. Outra alternativa é `SESSION-NAME`, como vimos acima. Se houver apenas uma sessão desanexada, nenhuma dessas opções é obrigatória:

```
$ screen -r
```

Este comando basta para anexar novamente a segunda sessão:

```
$ screen -ls
There is a screen on:
    1090.second session        (08/24/19 14:38:35)  (Attached)
1 Socket in /run/screen/S-carol.
```

Opções importantes para reanexar sessões:

-d -r

Reanexa uma sessão e — se necessário — a desanexa primeiro.

-d -R

Igual a `-d -r`, mas o `screen` cria primeiro uma sessão caso ela não exista.

-d -RR

Igual a `-d -R`. No entanto, usa a primeira sessão se houver mais de uma disponível.

-D -r

Reanexa uma sessão. Se necessário, o comando a desanexa e faz logout remotamente primeiro.

-D -R

Se uma sessão estiver rodando, ela é reanexada (desanexando e desconectando remotamente primeiro, caso necessário). Se não for o caso, ela é criada e o usuário, notificado.

-D -RR

Igual a `-D -R`— só que mais poderoso.

-d -m

Inicia o `screen` em *modo detached* (desanexado). Uma nova sessão é criada, mas desanexada. Útil para scripts de inicialização do sistema.

-D -m

Igual a `-d -m`, mas não bifurca um novo processo. O comando é encerrado se a sessão terminar.

Leia as páginas de manual de `screen` para descobrir outras opções.

Copiar e colar: Modo de rolagem

O GNU Screen apresenta um modo de cópia ou *modo de rolagem*. Dentro dele, você pode mover o cursor na janela atual e através do conteúdo de seu histórico usando as setas do teclado. Você pode marcar o texto e copiá-lo nas janelas. As etapas a seguir são:

1. Entre no modo de cópia/rolagem: `Ctrl l + a - [`.
2. Vá para o início da parte do texto que deseja copiar usando as setas do teclado.
3. Marque o início do trecho de texto que deseja copiar: Espaço.
4. Vá para o final da parte do texto que deseja copiar usando as setas do teclado.
5. Marque o final do trecho de texto que deseja copiar: Espaço.
6. Vá para a janela de sua escolha e cole o texto: `Ctrl l + a -]`.

Personalização de screen

O arquivo de configuração de todo o sistema para `screen` é `/etc/screenrc`. Também é possível usar `~/.screenrc` no nível do usuário. O arquivo inclui quatro seções de configuração principais:

SCREEN SETTINGS

Você pode definir configurações gerais especificando a *diretiva* seguida por um espaço e o *valor*, como em: `defscrollback 1024`.

SCREEN KEYBINDINGS

Esta seção é bastante interessante, pois permite redefinir os atalhos de teclado que podem interferir no seu uso diário do terminal. Use a palavra-chave `bind` seguida por um espaço, o caractere a ser usado após o prefixo do comando, outro espaço e o comando, como em: `bind l kill` (esta configuração mudará a forma padrão de eliminar uma janela para `Ctrl l + a - l`).

Para exibir todos os atalhos de `screen`, digite `Ctrl l + a - ?` ou consulte a página de manual.

TIP

Claro, você também pode alterar o próprio prefixo do comando. Por exemplo, para ir de `Ctrl + a` a `Ctrl + b`, basta adicionar esta linha: escape `^Bb`.

TERMINAL SETTINGS

Esta seção inclui as configurações relacionadas aos tamanhos de janela de terminal e buffers – entre outros. Para habilitar o modo sem bloqueio para lidar melhor com conexões SSH instáveis, por exemplo, a seguinte configuração é usada: `defnonblock 5`.

STARTUP SCREENS

Podemos incluir comandos para que diversos programas sejam executados após a inicialização de `screen`; por exemplo `screen -t top top` (o `screen` abre uma janela chamada `top` com `top` dentro dela).

tmux

O `tmux` foi lançado em 2007. Embora muito semelhante ao `screen`, ele apresenta algumas diferenças notáveis:

- Modelo cliente-servidor: o servidor fornece uma série de sessões, cada uma das quais pode ter várias janelas anexadas a ele que podem, por sua vez, ser compartilhadas por vários clientes.
- Seleção interativa de sessões, janelas e clientes via menus.
- A mesma janela pode ser anexada a várias sessões.

Disponibilidade de atalhos do *vim* e do *Emacs*.

- Suporte para terminal UTF-8 e 256 cores.

Janelas

O `tmux` pode ser invocado simplesmente digitando `tmux` no prompt de comando. Você verá um prompt do shell e uma barra de status na parte inferior da janela:

```
[0] 0:bash*                               "debian" 18:53
27-Aug-19
```

Além do hostname, a hora e a data, a barra de status fornece as seguintes informações:

Nome da sessão

`[0]`

Número da janela

0:

Nome da janela

bash*. Por padrão, esse é o nome do programa em execução dentro da janela e—ao contrário de screen—o tmux irá atualizá-lo automaticamente para refletir o programa em execução atual. Observe o asterisco indicando que esta é a janela atual em exibição.

Você pode atribuir nomes de sessão e janela ao invocar tmux:

```
$ tmux new -s "LPI" -n "Window zero"
```

A barra de status será alterada de acordo:

```
[LPI] 0:Window zero*                               "debian" 19:01
27-Aug-19
```

O prefixo do comando do tmux é **Ctrl L + b**. Para criar uma nova janela, basta digitar **Ctrl L + b - c**; você será levado a um novo prompt e a barra de status refletirá a nova janela:

```
[LPI] 0:Window zero- 1:bash*                         "debian" 19:02
27-Aug-19
```

Como o Bash é o shell subjacente, a nova janela recebe esse nome por padrão. Inicie top e veja como o nome muda para top:

```
[LPI] 0:Window zero- 1:top*                           "debian" 19:03
27-Aug-19
```

Em qualquer caso, você pode renomear uma janela com **Ctrl L + b - r**. Quando solicitado, informe o novo nome e pressione Enter:

```
(rename-window) Window one
```

Para exibir todas as janelas, pressione **Ctrl L + b - w** (use as setas do teclado para se mover para cima e para baixo e **enter** para selecionar):

```
(0) 0: Window zero- "debian"
(1) 1: Window one* "debian"
```

Como em `screen`, podemos pular de uma janela para outra com:

Ctrl + b - n

ir para a próxima janela, *next* em inglês.

Ctrl + b - p

ir para a janela anterior, *previous*.

Ctrl + b - number

ir para a janela de número *número*.

Para eliminar uma janela, use **Ctrl + b - &**. Será solicitada a confirmação:

```
kill-window Window one? (y/n)
```

Outros comandos de janela interessantes são:

Ctrl + b - f

encontrar uma janela pelo nome.

Ctrl + b - .

alterar o número do índice da janela.

Para conhecer a lista completa de comandos, consulte a página do manual.

Painéis

A facilidade de divisão de janelas do `screen` também está presente no `tmux`. As divisões resultantes não são chamadas de *regiões*, mas de *painéis*. A diferença mais importante entre regiões e painéis é que os últimos são pseudoterminais completos anexados a uma janela. Isso significa que eliminar um painel também eliminará seu pseudo-terminal e todos os programas associados em execução nele.

Para dividir uma janela horizontalmente, usamos **Ctrl + b - "**:

```
Tasks: 93 total, 1 running, 92 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
```

Memory Usage									Swap Usage	
Total			Free			Used			Available	Used
KiB	Mem		KiB	Mem		KiB	Mem			
4050960	total,		3730920	free,		114880	used,		205160	buff/cache
4192252	total,		4192252	free,		0	used.		3716004	avail Mem
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
1340	carol	20	0	44876	3400	2800	R	0.3	0.1	0:00.24 top
1	root	20	0	139088	6988	5264	S	0.0	0.2	0:00.50 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00 kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.04 ksoftirqd/0
4	root	20	0	0	0	0	S	0.0	0.0	0:01.62 kworker/0:0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 kworker/0:0H
7	root	20	0	0	0	0	S	0.0	0.0	0:00.06 rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00 rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00 migration/0
10	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 lru-add-drain
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.01 watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00 cpuhp/0

\$

\$

[LPI] 0:Window zero- 1:Window one*
19:05 27-Aug-19

"debian"

Para dividir la verticalmente, usamos **Ctrl + b - %**:

1

1 root 20 0 139088 6988 5264 5 0 0 0 2 0:00 50 systemd

1

1

1

```

5 root      0 -20      0      0      0 S 0.0 0.0 0:00.00 kworker/0:0H
7 root      20  0      0      0      0 S 0.0 0.0 0:00.06 rcu_sched   □
8 root      20  0      0      0      0 S 0.0 0.0 0:00.00 rcu_bh    □
9 root      rt  0      0      0      0 S 0.0 0.0 0:00.00 migration/0 □
                                         □
10 root     0 -20      0      0      0 S 0.0 0.0 0:00.00 lru-add-drai
n                                         □
11 root     rt  0      0      0      0 S 0.0 0.0 0:00.01 watchdog/0 □
                                         □
12 root     20  0      0      0      0 S 0.0 0.0 0:00.00 cpuhp/0  □
                                         □
$                                         □
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
$                                         □

```

[LPI] 0:Window zero- 1:Window one* "debian"
19:05 27-Aug-19

Para destruir o painel atual (junto com o pseudoterminal rodando dentro dele e quaisquer programas associados), use **Ctrl l + b -x**. Será solicitada a confirmação na barra de status:

kill-pane 1? (y/n)

Comandos importantes dos painéis:

Ctrl + b -↑, ↓, ←, →

mover-se entre painéis.

Ctrl + b - ;

passar para o último painel ativo.

Ctrl + b - Ctrl + seta

redimensionar o painel em uma linha.

Ctrl + b - Alt + seta

redimensionar o painel em cinco linhas.

Ctrl + b - {

trocar de painel (do atual para o anterior).

Ctrl + b - }

trocar de painel (do atual para o seguinte).

Ctrl + b - z

aproximar/afastar o painel.

Ctrl + b - t

O tmux exibe um relógio elegante dentro do painel (para removê-lo, pressione q).

Ctrl + b - !

transforma o painel em janela.

Para conhecer a lista completa de comandos, consulte a página do manual.

Sessões

Para listar as sessões no tmux, você pode usar **Ctrl + b - s**:

```
(0) + LPI: 2 windows (attached)
```

Outra alternativa é usar o comando **tmux ls**:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39] (attached)
```

Existe apenas uma sessão (LPI) que inclui duas janelas. Vamos criar uma nova sessão de dentro de

nossa sessão atual. Isso pode ser feito usando `Ctrl l + b`. Digite :new no prompt e pressione Enter. Você será enviado para a nova sessão, conforme pode ser observado na barra de status:

```
[2] 0: bash*                                "debian" 19:15
27-Aug-19
```

Por padrão, o tmux denomina a sessão 2. Para renomeá-la, use `Ctrl l + b - $`. Quando solicitado, informe o novo nome e pressione Enter:

```
(rename-session) Second Session
```

Para trocar de sessão, o atalho é `Ctrl l + b - s` (use as setas do teclado e `enter`):

```
(0) + LPI: 2 windows
(1) + Second Session: 1 windows (attached)
```

Para eliminar uma sessão, você pode usar o comando `tmux kill-session -t SESSION-NAME`. Se digitar o comando de dentro da sessão atual anexada, você será retirado do tmux e levado de volta à sua sessão de terminal inicial:

```
$ tmux kill-session -t "Second Session"
[exited]
$
```

Desanexando sessões

Ao eliminar `Second Session`, fomos levados para fora do tmux. No entanto, ainda temos uma sessão ativa. Peça ao tmux uma lista de sessões e você certamente a encontrará ali:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39]
```

No entanto, esta sessão está desanexada de seu terminal. Podemos anexá-la com `tmux attach -t SESSION-NAME` (`attach` pode ser substituído por `at` ou — simplesmente — `a`). Quando há apenas uma sessão, a especificação do nome é opcional:

```
$ tmux a
```

Agora você está de volta à sua sessão; para desanexá-la, pressione **Ctrl + b - d**:

```
[detached (from session LPI)]
$
```

TIP A mesma sessão pode ser anexada a mais de um terminal. Se quiser anexar uma sessão e ter certeza de que ela foi primeiramente desanexada de quaisquer outros terminais, use a opção `-d: tmux attach -d -t SESSION-NAME`.

Comandos importantes para anexar/desanexar sessão:

Ctrl + b - D

seleciona o cliente a desanexar.

Ctrl + b - r

atualiza o terminal do cliente.

Para conhecer a lista completa de comandos, consulte a página do manual.

Copiar e colar: Modo de rolagem

O tmux também possui um modo de cópia, basicamente igual ao do screen (lembre-se de usar o prefixo de comando do tmux e não o de screen!). A única diferença em termos de comando é que usamos **Ctrl + Espaço** para marcar o início da seleção e **Alt + w** para copiar o texto selecionado.

Personalização do tmux

Os arquivos de configuração do tmux tipicamente se localizam em `/etc/tmux.conf` and `~/.tmux.conf`. Quando iniciado, o tmux procura por esses arquivos, se eles existirem. Também existe a possibilidade de iniciar o tmux com a opção `-f` para fornecer um arquivo de configuração alternativo. Um exemplo de arquivo de configuração do tmux pode ser encontrado em `/usr/share/doc/tmux/example_tmux.conf`. O nível de personalização é altíssimo. Eis algumas das coisas que é possível fazer:

- Alterar a tecla de prefixo

```
# Change the prefix key to C-a
```

```
set -g prefix C-a
unbind C-b
bind C-a send-prefix
```

- Definir atalhos de teclado extras para janelas superiores a 9

```
# Some extra key bindings to select higher numbered windows
bind F1 selectw -t:10
bind F2 selectw -t:11
bind F3 selectw -t:12
```

Para ver uma lista abrangente com todos os atalhos, digite `Ctrl + b - ?` (pressione q para sair) ou consulte a página de manual.

Exercícios Guiados

1. Indique se as seguintes afirmações/recursos correspondem ao GNU Screen, ao tmux ou a ambos:

Recurso/Afirmação	GNU Screen	tmux
O prefixo dos comandos por padrão é <code>Ctrl + a</code>		
Modelo Cliente-Servidor		
Os painéis são pseudo-terminalis		
Eliminar uma região não elimina as janelas associadas		
As sessões incluem janelas		
As sessões podem ser desanexadas		

Mude o nome da janela atual para `vi`: . Instale o tmux em seu computador (nome do pacote: `tmux`) e realize as tarefas a seguir:

+ * Inicie o programa. Que comando você usa?

+

+ * Inicie `top` (note como – em poucos segundos – o nome da janela na barra de status muda para `top`):

+

+ * Usando o prefixo de teclado do tmux, abra uma nova janela; em seguida, crie `~/.tmux.conf` usando `nano`:

+

+ * Divida a janela verticalmente e reduza o tamanho do novo painel algumas vezes:

+

--	--	--

+ * Agora mude o nome da janela atual para `text editing`; em seguida, faça o `tmux` exibir uma lista de todas as sessões:

+

--	--	--

+ * Passe para a janela que está rodando `top` e volte à janela atual usando a mesma combinação de teclas:

+

--	--	--

+ * Desanexe a sessão atual e crie uma nova chamada `ssh` com uma janela chamada `ssh window`:

+

--	--	--

+ * Desanexe também a sessão `ssh` e faça o `tmux` exibir outra vez a lista de sessões:

+

--	--	--

+

A partir deste ponto, o exercício requer o uso de uma máquina *remota* para conexões `ssh` ao seu host local (uma máquina virtual é perfeitamente válida e pode ser muito prática). Certifique-se de ter o `openssh-server` instalado e rodando em sua máquina local e que pelo menos o `openssh-client` esteja instalado na máquina remota.

NOTE

+ * Agora, inicie uma máquina remota e conecte-se via `ssh` de seu host local. Quando a conexão for estabelecida, procure por sessões do `tmux`:

+

--	--	--

+ * No host remoto, anexe a sessão do ssh pelo nome:

+

+ * De volta à máquina local, anexe a sessão do ssh pelo nome, garantindo que a conexão ao hospedeiro remoto seja encerrada antes:

+

+ * Exiba todas as sessões para seleção e vá à primeira sessão ([0]). Uma vez ali, elimine a sessão ssh pelo nome:

+

+ * Finalmente, desanexe a sessão atual e elimine-a pelo nome:

+

Exercícios Exploratórios

1. Tanto `screen` quanto `tmux` podem entrar no modo de linha de comando através do atalho *prefixo do comando + :* (já vimos um breve exemplo com `tmux`). Faça uma pesquisa e realize as seguintes tarefas em modo de linha de comando:

- Faça o `screen` entrar em modo de cópia:

- Faça o `tmux` renomear a janela atual:

- Faça o `screen` fechar todas as janelas e encerrar a sessão:

- Faça o `tmux` dividir um painel em dois:

- Faça o `tmux` eliminar a janela atual:

2. Quando entramos no modo de cópia no `screen`, não somente podemos usar as setas do teclado e `PgUP` or `PgDown` para navegar na janela atual e no buffer de rolagem; também há a possibilidade de usar um editor de tela completo como o `vi`. Usando esse editor, realize as seguintes tarefas:

- Ecoe (echo) `supercalifragilisticexpialidocious` em seu terminal de `screen`:

- Agora, copie os cinco caracteres consecutivos (da esquerda para a direita) na linha acima de seu cursor:

- Finalmente, cole a seleção (`sticx`) em seu prompt de comando:

3. Suponha que você quer compartilhar uma sessão do `tmux` (`our_session`) com outro usuário. Você criou o socket (`/tmp/our_socket`) com as permissões corretas, para que você e o outro usuário

possam ler e escrever. Quais são as outras duas condições para que o segundo usuário possa anexar com sucesso a sessão usando tmux -S /tmp/our_socket a -t our_session?

Resumo

Nesta lição, você aprendeu sobre *multiplexadores de terminal* em geral e o GNU Screen e o tmux em particular. Estes são os conceitos importantes a lembrar:

- Prefixo do comando: o screen usa `Ctrl + a` + *caractere*; o tmux, `Ctrl + b` + *caractere*.
- Estrutura de sessões, janelas e divisões de janelas (regiões ou painéis).
- Modo de cópia.
- Desanexar sessões: um dos recursos mais poderosos dos multiplexadores.

Comandos usados nesta lição:

screen

Inicia uma sessão do screen.

tmux

Inicia uma sessão do tmux.

Respostas aos Exercícios Guiados

1. Indique se as seguintes afirmações/recursos correspondem ao GNU Screen, ao tmux ou a ambos:

Recurso/Afirmação	GNU Screen	tmux
O prefixo dos comandos por padrão é <code>Ctrl + a</code>	x	
Modelo Cliente-Servidor		x
Os painéis são pseudo-terminalis		x
Eliminar uma região não elimina as janelas associadas	x	
As sessões incluem janelas	x	x
As sessões podem ser desanexadas	x	x

2. Instale o GNU Screen no seu computador (nome do pacote: `screen`) e realize as seguintes tarefas:

- Inicie o programa. Que comando você usa?

`screen`

- Inicie `top`:

`top`

- Usando o prefixo de teclado de screen, abra uma nova janela; em seguida, abra `/etc/screenrc` usando `vi`:

`Ctrl + a - c`

`sudo vi /etc/screenrc`

- Liste as janelas na parte de baixo da tela:

`Ctrl + a - w`

- Mude o nome da janela atual para `vi`:

`Ctrl + a - A`. Depois digitamos `vi` e pressionamos `enter`.

- Mude o nome da janela restante para `top`. Para isso, primeiro exiba uma lista de todas as janelas para poder navegar entre elas e selecionar a correta:

Primeiro, digitamos `Ctrl + a - "`. Em seguida usamos as setas do teclado para marcar a janela que diz `0 bash` e pressionamos `enter`. Finalmente, digitamos `Ctrl + a - A`, em seguida `top` e pressionamos `enter`.

- Confira se os nomes foram alterados exibindo o nome das janelas novamente na parte de baixo da tela:

`Ctrl + a - w`

- Agora, desanexe a sessão e faça o `screen` criar uma nova de nome `ssh`:

`Ctrl + a - d screen -S "ssh"` e pressione `enter`.

- Desanexe `ssh` e faça o `screen` exibir a lista de sessões:

`Ctrl + a - d screen -list` ou `screen -ls`.

- Agora, anexe-a à primeira sessão usando seu PID:

`screen -r PID-OF-SESSION`

- Você deve ter retornado à janela que está exibindo `top`. Divida a janela horizontalmente e passe para a nova região vazia:

`Ctrl + a - S`

`Ctrl + a - Tab`

- Faça o `screen` listar todas as janelas e selecione `vi` para ser exibido na nova região vazia:

Usamos `Ctrl + a - "` para exibir todas as janelas para seleção, marcamos `vi` e pressionamos `enter`.

- Agora, divida a região atual verticalmente, passe para a nova região vazia e associe-a a uma nova janela:

`Ctrl + a - |`

`Ctrl + a - Tab`

Ctrl + a - c

- Elimine todas as regiões exceto a atual (lembre-se de que, embora você esteja eliminando as regiões, as janelas ainda estão vivas). Depois, saia de todas as janelas da sessão atual até que a sessão em si seja encerrada:

Ctrl + a - Q. `exit` (para sair do Bash). **Shift + :**, depois digitamos `quit` e pressionamos **enter** (para sair do `vi`). Em seguida, digitamos `exit` (para sair do shell Bash subjacente), `q` (para encerrar `top`); depois digitamos `exit` (para sair do shell Bash subjacente).

- Finalmente, faça o `screen` listar novamente suas sessões, eliminate a sessão `ssh` restante pelo PID e confira se não sobrou nenhuma sessão:

`screen -list ou screen -ls`

`screen -S PID-OF-SESSION -X quit`

`screen -list ou screen -ls`

3. Instale o `tmux` em seu computador (nome do pacote: `tmux`) e realize as tarefas a seguir:

- Inicie o programa. Que comando você usa?

`tmux`

- Inicie `top` (note como – em poucos segundos – o nome da janela na barra de status muda para `top`):

`top`

- Usando o prefixo de teclado do tmux, abra uma nova janela; em seguida, crie `~/.tmux.conf` usando `nano`:

Ctrl + b - c `nano ~/.tmux.conf`

- Divida a janela verticalmente e reduza o tamanho do novo painel algumas vezes:

Ctrl + b - "

Ctrl + b - Ctrl + ↓

- Agora mude o nome da janela atual para `text editing`; em seguida, faça o `tmux` exibir uma lista de todas as sessões:

`Ctrl + b - ,`. Daí informamos o novo nome e pressionamos `enter`. `Ctrl + b - s` ou `tmux ls`.

- Passe para a janela que está rodando `top` e volte à janela atual usando a mesma combinação de teclas:

`Ctrl + b - n` or `Ctrl + b - p`

- Desanexe a sessão atual e crie uma nova chamada `ssh` com uma janela chamada `ssh window`:

`Ctrl + b - d tmux new -s "ssh" -n "ssh window"`

- Desanexe também a sessão `ssh` e faça o `tmux` exibir outra vez a lista de sessões:

`Ctrl + b - d tmux ls`

NOTE

A partir deste ponto, o exercício requer o uso de uma máquina *remota* para conexões `ssh` ao seu host local (uma máquina virtual é perfeitamente válida e pode ser muito prática). Certifique-se de ter o `openssh-server` instalado e rodando em sua máquina local e que pelo menos o `openssh-client` esteja instalado na máquina remota.

- Agora, inicie uma máquina remota e conecte-se via `ssh` de seu host local. Quando a conexão for estabelecida, procure por sessões do `tmux`:

No hospedeiro remoto: `ssh local-username@local-ipaddress`. Depois de conectá-lo à máquina local: `tmux ls`.

- No host remoto, anexe a sessão do `ssh` pelo nome:

`tmux a -t ssh` (`a` pode ser substituído por `at` ou `attach`).

- De volta à máquina local, anexe a sessão do `ssh` pelo nome, garantindo que a conexão ao hospedeiro remoto seja encerrada antes:

`tmux a -d -t ssh` (`a` pode ser substituído por `at` ou `attach`).

- Exiba todas as sessões para seleção e vá à primeira sessão (`[0]`). Uma vez ali, elimine a sessão `ssh` pelo nome:

Digitamos `Ctrl + b - s`, usamos as setas do teclado para marcar a sessão `0` e damos `enter` `tmux kill-session -t ssh`.

- Finalmente, desanexe a sessão atual e elimine-a pelo nome:

```
Ctrl + b - d tmux kill-session -t 0.
```

Respostas aos Exercícios Exploratórios

1. Tanto screen quanto tmux podem entrar no modo de linha de comando através do atalho *prefixo do comando + :* (já vimos um breve exemplo com tmux). Faça uma pesquisa e realize as seguintes tarefas em modo de linha de comando:

- Faça o screen entrar em modo de cópia:

`Ctrl + a - :` — depois, digitamos copy.

- Faça o tmux renomear a janela atual:

`Ctrl + b - :` — depois, digitamos rename-window.

- Faça o screen fechar todas as janelas e encerrar a sessão:

`Ctrl + a - :` — depois, digitamos quit.

- Faça o tmux dividir um painel em dois:

`Ctrl + b - :` — depois, digitamos split-window.

- Faça o tmux eliminar a janela atual:

`Ctrl + b - :` — depois, digitamos kill-window.

2. Quando entramos no modo de cópia no screen, não somente podemos usar as setas do teclado e PgUP or PgDown para navegar na janela atual e no buffer de rolagem; também há a possibilidade de usar um editor de tela completo como o vi. Usando esse editor, realize as seguintes tarefas:

- Ecoe (echo) supercalifragilisticexpialidocious em seu terminal de screen:

```
echo supercalifragilisticexpialidocious
```

- Agora, copie os cinco caracteres consecutivos (da esquerda para a direita) na linha acima de seu cursor:

Para entrar no modo de cópia: `Ctrl + a - [` ou `Ctrl + a - :` e digitamos copy. Daí, passamos para a linha de cima usando `k` e pressionamos `espaço` para marcar o início da seleção. Finalmente, avançamos quatro caracteres usando `l` e pressionamos `espaço` novamente para marcar o final da seleção.

- Finalmente, cole a seleção (`sticx`) em seu prompt de comando:

[Ctrl l + a -]

3. Suponha que você quer compartilhar uma sessão do tmux (our_session) com outro usuário. Você criou o socket (/tmp/our_socket) com as permissões corretas, para que você e o outro usuário possam ler e escrever. Quais são as outras duas condições para que o segundo usuário possa anexar com sucesso a sessão usando tmux -S /tmp/our_socket a -t our_session?

Ambos os usuários devem ter um grupo em comum, p. ex. multiplexer. Em seguida o socket também deve ser passado para esse grupo: chgrp multiplexer /tmp/our_socket.



103.6 Modificar a prioridade de execução de um processo

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.6

Peso

2

Áreas chave de conhecimento

- Saber a prioridade padrão de um processo que é criado.
- Executar um programa com maior ou menor prioridade do que o padrão.
- Mudar a prioridade de um processo em execução.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `nice`
- `ps`
- `renice`
- `top`



103.6 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.6 Modificar prioridades de execução de processos
Lição:	1 de 1

Introdução

Os sistemas operacionais capazes de executar mais de um processo ao mesmo tempo são chamados de sistemas multitarefa ou multiprocessamento. Embora a verdadeira simultaneidade só aconteça quando há mais de uma unidade de processamento disponível, os sistemas de processador único podem imitar a simultaneidade alternando entre os processos muito rapidamente. Essa técnica também é empregada em sistemas com muitas CPUs equivalentes, ou sistemas de *multiprocessamento simétrico (SMP)*, dado que o número de processos concorrentes em potencial excede em muito o número de unidades de processamento disponíveis.

Na verdade, apenas um processo por vez pode controlar a CPU. No entanto, a maioria das atividades processadas são *chamadas do sistema*, ou seja, o processo em execução transfere o controle da CPU para um processo do sistema operacional para que ele execute a operação solicitada. As chamadas do sistema são responsáveis por toda a comunicação entre dispositivos, como alocações de memória, leitura e gravação em sistemas de arquivos, impressão de texto na tela, interação com o usuário, transferências de rede, etc. Transferir o controle da CPU durante uma chamada do sistema permite que o sistema operacional decida se deve devolver o controle da CPU para o processo anterior ou transferi-lo para outro processo. Como as CPUs modernas são capazes de executar instruções muito

mais rápido do que a maioria dos hardwares externos pode se comunicar entre si, um novo processo de controle pode fazer boa parte do trabalho da CPU enquanto as respostas de hardware solicitadas anteriormente ainda não estão disponíveis. Para garantir o aproveitamento máximo da CPU, os sistemas operacionais de multiprocessamento mantêm uma fila dinâmica de processos ativos aguardando um slot de tempo da CPU.

Embora elas permitam melhorar significativamente a utilização do tempo da CPU, confiar apenas nas chamadas do sistema para alternar entre os processos não basta para obter um desempenho multitarefa satisfatório. Um processo que não faz chamadas de sistema poderia controlar a CPU indefinidamente. Por essa razão, os sistemas operacionais modernos também são *preventivos*, ou seja, um processo em execução pode ser posto de volta na fila para que um processo mais importante assuma o controle da CPU, mesmo que o processo em execução não tenha feito uma chamada de sistema.

O Agendador do Linux

O Linux, sendo um sistema operacional de multiprocessamento preventivo, implementa um agendador que organiza a fila de processos. Mais precisamente, o agendador também decide qual *thread* da fila será executada — um processo pode ter muitas threads independentes — mas processo e thread são termos intercambiáveis neste contexto. Cada processo tem dois predicados que intervêm em seu agendamento: a *política de programação* e a *prioridade de programação*.

Existem dois tipos principais de políticas de programação: *políticas em tempo real* e *políticas normais*. Os processos em uma política em tempo real são programados diretamente por seus valores de prioridade. Se um processo mais importante estiver pronto para ser executado, um processo menos importante será interrompido e o processo de prioridade mais alta assumirá o controle da CPU. Um processo de prioridade mais baixa obterá o controle da CPU apenas se os processos de prioridade mais alta estiverem ociosos ou aguardando a resposta do hardware.

Qualquer processo em tempo real tem mais prioridade do que um processo normal. Como um sistema operacional de uso geral, o Linux executa apenas um punhado de processos em tempo real. A maioria dos processos, incluindo o sistema e os programas, são executados sob as políticas de programação normais. Os processos normais geralmente têm o mesmo valor de prioridade, mas as políticas normais podem definir regras de prioridade de execução usando outro predicado do processo: o *valor nice*. Para evitar confusão com as prioridades dinâmicas derivadas de valores nice, as prioridades de programação são geralmente chamadas de prioridades *estáticas*.

O agendador do Linux pode ser configurado de muitas maneiras diferentes e existem formas ainda mais intrincadas de estabelecer prioridades, mas esses conceitos gerais sempre se aplicam. Ao inspecionar e refinar a programação do processo, é importante ter em mente que apenas os processos sob a política de programação normal serão afetados.

Como ler as prioridades

O Linux reserva as prioridades estáticas de 0 a 99 para processos em tempo real. As prioridades estáticas de 100 a 139 são atribuídas a processos normais, o que significa que existem 39 níveis de prioridade diferentes para os processos normais. Valores mais baixos indicam uma prioridade mais alta. A prioridade estática de um processo ativo pode ser encontrada no arquivo `sched`, localizado em seu diretório respectivo dentro do sistema de arquivos `/proc`:

```
$ grep ^prio /proc/1/sched
prio : 120
```

Como mostrado no exemplo, a linha que começa com `prio` fornece o valor de prioridade do processo (o processo PID 1 é o processo `init` ou `systemd`, o primeiro processo que o kernel inicia durante a inicialização do sistema). A prioridade padrão para os processos normais é 120, podendo assim ser diminuída para 100 ou aumentada para 139. As prioridades de todos os processos em execução podem ser verificadas com o comando `ps -Al` ou `ps -el`:

\$ ps -el													
F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	0	1	0	0	80	0	-	9292	-	?	00:00:00	systemd
4	S	0	19	1	0	80	0	-	8817	-	?	00:00:00	systemd-journal
4	S	104	61	1	0	80	0	-	64097	-	?	00:00:00	rsyslogd
4	S	0	63	1	0	80	0	-	7244	-	?	00:00:00	cron
1	S	0	126	1	0	80	0	-	4031	-	?	00:00:00	dhclient
4	S	0	154	1	0	80	0	-	3937	-	pts/0	00:00:00	agetty
4	S	0	155	1	0	80	0	-	3937	-	pts/1	00:00:00	agetty
4	S	0	156	1	0	80	0	-	3937	-	pts/2	00:00:00	agetty
4	S	0	157	1	0	80	0	-	3937	-	pts/3	00:00:00	agetty
4	S	0	158	1	0	80	0	-	3937	-	console	00:00:00	agetty
4	S	0	160	1	0	80	0	-	16377	-	?	00:00:00	sshd
4	S	0	280	0	0	80	0	-	5301	-	?	00:00:00	bash
0	R	0	392	280	0	80	0	-	7221	-	?	00:00:00	ps

A coluna `PRI` indica a prioridade estática atribuída pelo kernel. Observe, no entanto, que o valor de prioridade exibido por `ps` difere do obtido no exemplo anterior. Por razões históricas, as prioridades exibidas por `ps` variam de -40 a 99 por padrão, de modo que a prioridade real é obtida adicionando 40 a esse valor (no caso, $80 + 40 = 120$).

Também é possível monitorar continuamente os processos atualmente gerenciados pelo kernel do Linux com o programa `top`. Como `ps`, `top` também exibe o valor de prioridade de forma diferente. Para facilitar a identificação de processos em tempo real, `top` subtrai 100 do valor da prioridade,

tornando todas as prioridades em tempo real negativas, com um número negativo ou *rt* que as identifica. Portanto, as prioridades normais exibidas por `top` variam de 0 a 39.

Para obter mais detalhes do comando `ps`, podemos usar opções adicionais. Compare a saída deste comando com a de nosso exemplo anterior:

NOTE

```
$ ps -e -o user,uid,comm,tty,pid,ppid,pri,pmem,pcpu --sort=-pcpu | head
```

Valor nice

Todo processo normal começa com um valor nice padrão de 0 (prioridade 120). O nome *nice* (agradável, cortês) vem da ideia de que os processos “mais corteses” permitem que outros processos sejam executados antes deles em uma fila de execução particular. Os números nice variam de -20 (menos cortês, prioridade alta) a 19 (mais cortês, prioridade baixa). O Linux também permite atribuir diferentes valores nice a threads do mesmo processo. A coluna NI na saída de `ps` indica o número *nice*.

Apenas o usuário root pode diminuir o valor nice de um processo para menos de zero. É possível iniciar um processo com uma prioridade não padrão com o comando `nice`. Por padrão, `nice` muda o valor nice para 10, mas esse valor pode ser especificado com a opção `-n`:

```
$ nice -n 15 tar czf home_backup.tar.gz /home
```

Neste exemplo, o comando `tar` é executado com um valor nice de 15. O comando `renice` pode ser usado para alterar a prioridade de um processo em execução. A opção `-p` indica o número PID do processo alvo. Por exemplo:

```
# renice -10 -p 2164
2164 (process ID) old priority 0, new priority -10
```

As opções `-g` e `-u` são usadas para modificar todos os processos de um determinado grupo ou usuário, respectivamente. Com `renice +5 -g users`, o valor nice dos processos pertencentes a usuários do grupo *users* será aumentado em cinco.

Além de `renice`, a prioridade dos processos pode ser modificada com outros programas, como `top`. No alto da tela principal, o valor nice de um processo pode ser modificado pressionando `r` e, em seguida, o número PID do processo:

```

top - 11:55:21 up 23:38, 1 user, load average: 0,10, 0,04, 0,05
Tasks: 20 total, 1 running, 19 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,5 us, 0,3 sy, 0,0 ni, 99,0 id, 0,0 wa, 0,2 hi, 0,0 si, 0,0 st
KiB Mem : 4035808 total, 774700 free, 1612600 used, 1648508 buff/cache
KiB Swap: 7999828 total, 7738780 free, 261048 used. 2006688 avail Mem
PID to renice [default pid = 1]
  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
    1 root      20   0  74232  7904  6416 S 0,000 0,196  0:00.12 systemd
    15 root     20   0  67436  6144  5568 S 0,000 0,152  0:00.03 systemd-
journal
   21 root     20   0  61552  5628  5000 S 0,000 0,139  0:00.01 systemd-logind
   22 message+ 20   0  43540  4072  3620 S 0,000 0,101  0:00.03 dbus-daemon
   23 root     20   0  45652  6204  4992 S 0,000 0,154  0:00.06 wickedd-dhcp4
   24 root     20   0  45648  6276  5068 S 0,000 0,156  0:00.06 wickedd-auto4
   25 root     20   0  45648  6272  5060 S 0,000 0,155  0:00.06 wickedd-dhcp6

```

A mensagem `PID to renice [default pid = 1]` aparece, com o primeiro processo listado selecionado por padrão. Para alterar a prioridade de outro processo, digite o PID dele e pressione Enter. A mensagem `Renice PID 1 to value` será exibida (com o número PID solicitado) e um novo valor nice poderá ser atribuído.

Exercícios Guiados

1. Em um sistema multitarefa preventivo, o que acontece quando um processo de prioridade mais baixa está ocupando o processador e um processo de prioridade mais alta é posto na fila para ser executado?

2. Considere a seguinte tela de top:

```
top - 08:43:14 up 23 days, 12:29, 5 users, load average: 0,13, 0,18, 0,21
Tasks: 240 total, 2 running, 238 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1,4 us, 0,4 sy, 0,0 ni, 98,1 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 7726,4 total, 590,9 free, 1600,8 used, 5534,7 buff/cache
MiB Swap: 30517,0 total, 30462,5 free, 54,5 used. 5769,4 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 1 root 20 0 171420 10668 7612 S 0,0 0,1 9:59.15 systemd
 2 root 20 0 0 0 0 S 0,0 0,0 0:02.76 kthreadd
 3 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 rcu_gp
 4 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 rcu_par_gp
 8 root 0 -20 0 0 0 I 0,0 0,0 0:00.00
mm_percpu_wq
 9 root 20 0 0 0 0 S 0,0 0,0 0:49.06
ksoftirqd/0
 10 root 20 0 0 0 I 0,0 0,0 18:24.20 rcu_sched
 11 root 20 0 0 0 I 0,0 0,0 0:00.00 rcu_bh
 12 root rt 0 0 0 0 S 0,0 0,0 0:08.17
migration/0
 14 root 20 0 0 0 S 0,0 0,0 0:00.00 cpuhp/0
 15 root 20 0 0 0 S 0,0 0,0 0:00.00 cpuhp/1
 16 root rt 0 0 0 0 S 0,0 0,0 0:11.79
migration/1
 17 root 20 0 0 0 0 S 0,0 0,0 0:26.01
ksoftirqd/1
```

Quais PIDs têm prioridades em tempo real?

3. Considere a seguinte listagem de ps -el:

F S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4 S	0	1	0	0	80	0	-	42855	-	?	00:09:59	systemd

1 S	0	2	0	0	80	0	-	0	-	?	00:00:02	kthreadd
1 I	0	3	2	0	60	-20	-	0	-	?	00:00:00	rcu_gp
1 S	0	9	2	0	80	0	-	0	-	?	00:00:49	ksoftirqd/0
1 I	0	10	2	0	80	0	-	0	-	?	00:18:26	rcu_sched
1 I	0	11	2	0	80	0	-	0	-	?	00:00:00	rcu_bh
1 S	0	12	2	0	-40	--	--	0	-	?	00:00:08	migration/0
1 S	0	14	2	0	80	0	-	0	-	?	00:00:00	cpuhp/0
5 S	0	15	2	0	80	0	-	0	-	?	00:00:00	cpuhp/1

Qual PID tem a prioridade mais alta?

4. Depois de tentar alterar o valor nice de um processo com `renice`, acontece o erro a seguir:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Qual a causa provável desse erro?

Exercícios Exploratórios

1. A alteração da prioridades dos processos geralmente é necessária quando um processo está ocupando muito tempo da CPU. Usando `ps` com opções padrão para imprimir todos os processos do sistema em formato longo, qual sinalizador de `--sort` permite classificar os processos por utilização da CPU, em ordem crescente?

2. O comando `schedtool` pode definir todos os parâmetros de agendamento da CPU de que o Linux é capaz ou exibir informações para determinados processos. Como ele pode ser usado para exibir os parâmetros de agendamento do processo 1750? Além disso, como `schedtool` pode ser usado para alterar o processo 1750 para tempo real com prioridade -90 (conforme exibido por `top`)?

Resumo

Esta lição trata de como o Linux compartilha o tempo da CPU entre os processos gerenciados. Para garantir o melhor desempenho, os processos mais críticos devem ter prioridade sobre os menos críticos. A lição explica as seguintes etapas:

- Conceitos básicos sobre sistemas de multiprocessamento.
- O que é um agendador de processos e como o Linux o implementa.
- O que são as prioridades no Linux, números nice e sua finalidade.
- Como ler e interpretar prioridades de processos no Linux.
- Como alterar a prioridade de um processo antes e durante sua execução.

Respostas aos Exercícios Guiados

1. Em um sistema multitarefa preventivo, o que acontece quando um processo de prioridade mais baixa está ocupando o processador e um processo de prioridade mais alta é posto na fila para ser executado?

O processo de prioridade mais baixa é pausado e o de prioridade mais alta é executado em seu lugar.

2. Considere a seguinte tela de top:

```
top - 08:43:14 up 23 days, 12:29, 5 users, load average: 0,13, 0,18, 0,21
Tasks: 240 total, 2 running, 238 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1,4 us, 0,4 sy, 0,0 ni, 98,1 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 7726,4 total, 590,9 free, 1600,8 used, 5534,7 buff/cache
MiB Swap: 30517,0 total, 30462,5 free, 54,5 used. 5769,4 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
        1 root      20   0 171420 10668  7612 S  0,0  0,1  9:59.15 systemd
        2 root      20   0          0      0      0 S  0,0  0,0  0:02.76 kthreadd
        3 root      0 -20          0      0      0 I  0,0  0,0  0:00.00 rcu_gp
        4 root      0 -20          0      0      0 I  0,0  0,0  0:00.00 rcu_par_gp
        8 root      0 -20          0      0      0 I  0,0  0,0  0:00.00
mm_percpu_wq
        9 root      20   0          0      0      0 S  0,0  0,0  0:49.06
ksoftirqd/0
       10 root      20   0          0      0      0 I  0,0  0,0  18:24.20 rcu_sched
       11 root      20   0          0      0      0 I  0,0  0,0  0:00.00 rcu_bh
       12 root      rt   0          0      0      0 S  0,0  0,0  0:08.17
migration/0
       14 root      20   0          0      0      0 S  0,0  0,0  0:00.00 cpuhp/0
       15 root      20   0          0      0      0 S  0,0  0,0  0:00.00 cpuhp/1
       16 root      rt   0          0      0      0 S  0,0  0,0  0:11.79
migration/1
       17 root      20   0          0      0      0 S  0,0  0,0  0:26.01
ksoftirqd/1
```

Quais PIDs têm prioridades em tempo real?

PIDs 12 and 16.

3. Considere a seguinte listagem de ps -el:

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	0	1	0	0	80	0	-	42855	-	?	00:09:59	systemd
1	S	0	2	0	0	80	0	-	0	-	?	00:00:02	kthreadd
1	I	0	3	2	0	60	-20	-	0	-	?	00:00:00	rcu_gp
1	S	0	9	2	0	80	0	-	0	-	?	00:00:49	ksoftirqd/0
1	I	0	10	2	0	80	0	-	0	-	?	00:18:26	rcu_sched
1	I	0	11	2	0	80	0	-	0	-	?	00:00:00	rcu_bh
1	S	0	12	2	0	-40	--	-	0	-	?	00:00:08	migration/0
1	S	0	14	2	0	80	0	-	0	-	?	00:00:00	cpuhp/0
5	S	0	15	2	0	80	0	-	0	-	?	00:00:00	cpuhp/1

Qual PID tem a prioridade mais alta?

O PID 12.

4. Depois de tentar alterar o valor nice de um processo com `renice`, acontece o erro a seguir:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Qual a causa provável desse erro?

Apenas o usuário root pode definir os números nice para menos de zero.

Respostas aos Exercícios Exploratórios

1. A alteração da prioridades dos processos geralmente é necessária quando um processo está ocupando muito tempo da CPU. Usando `ps` com opções padrão para imprimir todos os processos do sistema em formato longo, qual sinalizador de `--sort` permite classificar os processos por utilização da CPU, em ordem crescente?

```
$ ps -el --sort=pcpu
```

2. O comando `schedtool` pode definir todos os parâmetros de agendamento da CPU de que o Linux é capaz ou exibir informações para determinados processos. Como ele pode ser usado para exibir os parâmetros de agendamento do processo 1750? Além disso, como `schedtool` pode ser usado para alterar o processo 1750 para tempo real com prioridade -90 (conforme exibido por `top`)?

```
$ schedtool 1750
```

```
$ schedtool -R -p 89 1750
```



103.7 Procurar em arquivos de texto usando expressões regulares

Referência ao LPI objectivo

[LPIC-1 v5, Exam 101, Objective 103.7](#)

Peso

3

Áreas chave de conhecimento

- Criar expressões regulares contendo vários elementos.
- Entender a diferença entre expressões regulares básicas e estendidas.
- Entender os conceitos de caracteres especiais, classes de caracteres, quantificadores e âncoras.
- Usar ferramentas de expressão regular para realizar pesquisas pelo sistema de arquivos ou no conteúdo de um arquivo.
- Utilizar expressões regulares para apagar, alterar e substituir texto.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- grep
- egrep
- fgrep
- sed
- regex(7)



103.7 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.7 Pesquisar em arquivos de texto usando expressões regulares
Lição:	1 de 2

Introdução

Os algoritmos de pesquisa de string são muito usados por diversas tarefas de processamento de dados, a tal ponto que os sistemas operacionais Unix têm sua própria implementação onipresente: as *expressões regulares*, freqüentemente abreviadas para *ERs*. As expressões regulares consistem em sequências de caracteres constituindo um padrão genérico usado para localizar e, às vezes, modificar uma sequência correspondente em uma sequência maior de caracteres. As expressões regulares expandem muito a capacidade de:

- Escrever regras de análise (parsing) para solicitações em servidores HTTP, *nginx* em particular.
- Escrever scripts que convertem conjuntos de dados baseados em texto para outro formato.
- Procurar por ocorrências de interesse em entradas de diário ou documentos.
- Filtrar documentos de marcação, mantendo o conteúdo semântico.

A expressão regular mais simples contém pelo menos um *átomo*. Um átomo, assim chamado por ser o elemento básico de uma expressão regular, é apenas um caractere que pode ou não ter um significado especial. A maioria dos caracteres comuns não são ambíguos e retêm seu significado

literal, enquanto outros têm um significado especial:

• (ponto)

O átomo corresponde a qualquer caractere.

^ (acento circunflexo)

O átomo corresponde ao início de uma linha.

\$ (cifrão)

O átomo corresponde ao fim de uma linha.

Por exemplo, a expressão regular `bc`, composta pelos átomos literais `b` e `c`, pode ser encontrada na string `abcd`, mas não na string `a1cd`. Por outro lado, a expressão regular `.c` pode ser encontrada nas strings `abcd` e `a1cd`, pois o ponto `.` corresponde a qualquer caractere.

Os átomos de circunflexo e o cífrão são usados quando apenas as correspondências no início ou no final da string são de interesse. Por essa razão, eles também são chamados de *âncoras*. Assim, `cd` pode ser encontrado em `abcd`, mas `^cd` não. Da mesma forma, `ab` pode ser encontrado em `abcd`, mas `ab$` não. O acento circunflexo `^` é um caractere literal, exceto quando no início, e `$` é um caractere literal, exceto quando no final da expressão regular.

Expressão de colchetes

Existe outro tipo de átomo denominado *expressão de colchetes*. Embora não sejam um único caractere, os colchetes `[]` (incluindo o conteúdo) são considerados um único átomo. Uma expressão de colchetes geralmente é apenas uma lista de caracteres literais delimitados por `[]`, fazendo com que o átomo corresponda a qualquer caractere único da lista. Por exemplo, a expressão `[1b]` pode ser encontrada nas strings `abcd` e `a1cd`. Para especificar caracteres aos quais o átomo não deve corresponder, a lista deve começar com `^`, como em `[^1b]`. Também é possível especificar intervalos de caracteres em expressões de colchetes. Por exemplo, `[0-9]` corresponde aos dígitos de 0 a 9 e `[a-z]` corresponde a qualquer letra minúscula. Os intervalos devem ser usados com cuidado, pois podem não ser consistentes em diferentes idiomas.

As listas de expressão de colchetes também aceitam classes em vez de apenas caracteres únicos e intervalos. As classes de caracteres tradicionais são:

`[:alnum:]`

Representa um caractere alfanumérico.

`[:alpha:]`

Representa um caractere alfabético.

[:asci:]

Representa um caractere que pertence ao conjunto de caracteres ASCII.

[:blank:]

Representa um caractere em branco, ou seja, um espaço ou tabulação.

[:cntrl:]

Representa um caractere de controle.

[:digit:]

Representa um dígito (de 0 a 9).

[:graph:]

Representa qualquer caractere imprimível, exceto espaço.

[:lower:]

Representa um caractere em minúsculas.

[:print:]

Representa qualquer caractere imprimível, incluindo espaço.

[:punct:]

Representa qualquer caractere imprimível que não seja um espaço ou um caractere alfanumérico.

[:space:]

Representa os caracteres de espaço em branco: espaço, alimentação de formulário (\f), nova linha (\n), retorno de carro (\r), tabulação horizontal (\t) e tabulação vertical (\v).

[:upper:]

Representa uma letra maiúscula.

[:xdigit:]

Representa dígitos hexadecimais (de 0 a F).

As classes de caracteres podem ser combinadas com caracteres únicos e intervalos, mas não podem ser usadas como ponto final de um intervalo. Além disso, as classes de caracteres podem ser usadas apenas em expressões de colchetes, não como um átomo independente fora dos colchetes.

Quantificadores

O alcance de um átomo, seja um átomo de caractere único ou um átomo de colchete, pode ser ajustado usando um *quantificador de átomo*. Os quantificadores de átomos definem *seqüências* de átomos, ou seja, as correspondências ocorrem quando uma repetição contígua para o átomo é encontrada na string. A substring que casa com a correspondência é chamada de *peça*. Não obstante, quantificadores e outros recursos de expressões regulares são tratados de maneira diferente dependendo do padrão que está sendo usado.

Conforme definido pelo POSIX, existem duas formas de expressões regulares: expressões regulares “básicas” e expressões regulares “estendidas”. A maioria dos programas que trabalham com texto, em qualquer distribuição Linux convencional, oferece suporte a ambas as formas, por isso é importante conhecer as diferenças para evitar problemas de compatibilidade e escolher a implementação mais adequada para a tarefa em questão.

O quantificador `*` tem a mesma função em ERs básicas e estendidas (o átomo ocorre zero ou mais vezes) e é um caractere literal se aparecer no início da expressão regular ou se for precedido por uma barra invertida `\`. O quantificador de sinal de mais `+` seleciona as peças que contêm uma ou mais correspondências de átomos em sequência. Com o quantificador de ponto de interrogação `?`, a correspondência ocorrerá se o átomo correspondente aparecer uma vez ou se não aparecer. Se precedido por uma barra invertida `\`, seu significado especial não é considerado. As expressões regulares básicas também suportam os quantificadores `+` e `?`, mas eles precisam ser precedidos por uma barra invertida. Ao contrário das expressões regulares estendidas, `+` e `?` sozinhos são caracteres literais em expressões regulares básicas.

Chaves

As *chaves* são quantificadores que permitem ao usuário especificar limites precisos de quantidade para um átomo. Em expressões regulares estendidas, as chaves podem aparecer de três maneiras:

{i}

O átomo deve aparecer exatamente `i` vezes (sendo `i` um número inteiro). Por exemplo, `[[[:blank:]]{2}]` corresponde a exatamente dois caracteres em branco.

{i,}

O átomo deve aparecer pelo menos `i` vezes (sendo `i` um número inteiro). Por exemplo, `[[[:blank:]]{2,}]` corresponde a qualquer sequência de dois ou mais caracteres em branco.

{i,j}

The atom must appear at least `i` times and at most `j` times (`i` and `j` integer numbers, `j` greater

then i). For example, xyz{2,4} matches the xy string followed by two to four of the z character.

De toda forma, se uma substring corresponder a uma expressão regular e uma substring mais longa começando no mesmo ponto também corresponder, a substring mais longa será considerada.

As expressões regulares básicas também suportam chaves, mas elas devem ser precedidas por \: \{ e \}. Sozinhas, { e } são interpretadas como caracteres literais. Uma \{ seguida por um caractere diferente de um dígito é um caractere literal, não uma abertura de chave.

Alternâncias e agrupamentos

As expressões regulares básicas também diferem das expressões regulares estendidas em outro aspecto importante: uma expressão regular estendida pode ser dividida em *alternâncias* (branches), sendo cada uma delas uma expressão regular independente. As alternativas são separadas por | e a expressão regular combinada corresponderá a qualquer coisa que corresponda a qualquer uma das alternativas. Por exemplo, he|him irá corresponder se a substring he ou a substring him forem encontradas na string que está sendo examinada. As expressões regulares básicas interpretam | como um caractere literal. No entanto, a maioria dos programas que suportam expressões regulares básicas permitem alternâncias com \|.

Uma expressão regular estendida entre () pode ser usada em um *agrupamento* (back reference). Por exemplo, ([[:digit:]])\1 corresponde a qualquer expressão regular que se repita pelo menos uma vez, porque o \1 na expressão é o agrupamento da peça encontrada pela primeira subexpressão entre parênteses. Se houver mais de uma subexpressão entre parênteses na expressão regular, elas podem ser referenciadas com \2, \3 e assim por diante.

Para ERs básicas, as subexpressões devem ser colocadas entre \(\(e \)\), sendo (and) sozinhos caracteres comuns. O indicador de agrupamento é usado como nas expressões regulares estendidas.

Pesquisas com expressões regulares

A vantagem imediata oferecida pelas expressões regulares é aprimorar as pesquisas em sistemas de arquivos e em documentos de texto. A opção `-regex` do comando `find` permite testar cada caminho em uma hierarquia de diretórios de acordo com uma expressão regular. Por exemplo,

```
$ find $HOME -regex '.*\..*' -size +100M
```

busca por arquivos maiores que 100 megabytes (100 unidades de 1048576 bytes), mas apenas em caminhos dentro do diretório inicial do usuário que contenham uma correspondência com .*\..*, ou seja, um /. rodeado por qualquer outro número de caracteres. Em outras palavras, apenas os arquivos ocultos ou arquivos dentro de diretórios ocultos serão listados, independentemente da

posição de `/.` no caminho correspondente. Para expressões regulares que não diferenciam maiúsculas de minúsculas, a opção `-iregex` deve ser usada em seu lugar:

```
$ find /usr/share/fonts -regextype posix-extended -iregex
'.*(dejavu|liberation).*sans.*(italic|oblique).*'
/usr/share/fonts/dejavu/DejaVuSansCondensed-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSansCondensed-Oblique.ttf
/usr/share/fonts/dejavu/DejaVuSans-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSans-Oblique.ttf
/usr/share/fonts/dejavu/DejaVuSansMono-BoldOblique.ttf
/usr/share/fonts/dejavu/DejaVuSansMono-Oblique.ttf
/usr/share/fonts/liberation/LiberationSans-BoldItalic.ttf
/usr/share/fonts/liberation/LiberationSans-Italic.ttf
```

Neste exemplo, a expressão regular contém alternâncias (escritas no estilo *estendido*) para listar apenas arquivos de fonte específicos na hierarquia de diretório `/usr/share/fonts`. Expressões regulares estendidas não são suportadas por padrão, mas `find` permite que sejam habilitadas com `-regextype posix-extended` or `-regextype egrep`. A ER padrão para `find` é `findutils-default`, que é essencialmente um clone da expressão regular básica.

Freqüentemente, é necessário passar a saída de um programa para o comando `less` quando ele não cabe na tela. O comando `less` divide a entrada em páginas, uma tela de cada vez, permitindo ao usuário navegar facilmente pelo texto para cima e para baixo. Além disso, `less` também permite que um usuário execute pesquisas baseadas em expressões regulares. Este recurso é notavelmente importante porque `less` é o paginador padrão usado para muitas tarefas diárias, como inspecionar entradas de diário ou consultar páginas de manual. Ao ler uma página de manual, por exemplo, pressionar a tecla `/` abre um prompt de pesquisa. Esse é um cenário típico em que as expressões regulares são úteis, pois as opções do comando são listadas logo após a margem da página no layout geral da página do manual. No entanto, a mesma opção pode aparecer muitas vezes no texto, inviabilizando as buscas literais. Independentemente disso, digitar `^[[:blank:]]*-o`—ou, mais simplesmente: `^ *-o`—no prompt de pesquisa permite pular imediatamente para a opção da seção `-o` (se existente) após pressionar Enter, permitindo assim consultar mais rapidamente uma descrição da opção.

Exercícios Guiados

1. Qual expressão regular estendida corresponderia a qualquer endereço de e-mail, como `info@example.org`?

2. Qual expressão regular estendida corresponderia apenas a qualquer endereço IPv4 no formato pontilhado-quad padrão, como `192.168.15.1`?

3. Como o comando `grep` pode ser usado para listar o conteúdo do arquivo `/etc/services`, descartando todos os comentários (linhas começando com `#`)?

4. O arquivo `domains.txt` contém uma lista de nomes de domínio, um por linha. Como o comando `egrep` seria usado para listar apenas domínios `.org` or `.com`?

Exercícios Exploratórios

1. A partir do diretório atual, como o comando `find` usaria uma expressão regular estendida para pesquisar todos os arquivos que não contêm um sufixo de arquivo padrão (nomes de arquivo que não terminam em `.txt` ou `.c`, por exemplo)?

2. O comando `less` é o paginador padrão para exibir arquivos de texto longos no ambiente shell. Ao digitar `/`, uma expressão regular pode ser inserida no prompt de pesquisa para pular para a primeira correspondência pertinente. Para permanecer na posição atual do documento e destacar apenas as correspondências pertinentes, qual combinação de teclas deve ser inserida no prompt de pesquisa?

3. Em `less`, como seria possível filtrar a saída para que apenas as linhas que correspondem a uma expressão regular sejam exibidas?

Resumo

Esta lição cobre o suporte geral do Linux para expressões regulares, um padrão amplamente usado cujos recursos de correspondência de padrões são suportados pela maioria dos programas que trabalham com texto. A lição atravessa as seguintes etapas:

- O que é uma expressão regular.
- Os principais componentes de uma expressão regular.
- As diferenças entre expressões regulares e expressões regulares estendidas.
- Como realizar pesquisas simples de texto e arquivos usando expressões regulares.

Respostas aos Exercícios Guiados

1. Qual expressão regular estendida corresponderia a qualquer endereço de e-mail, como info@example.org?

```
egrep "\S+@\S+\.\S+"
```

2. Qual expressão regular estendida corresponderia apenas a qualquer endereço IPv4 no formato pontilhado-quad padrão, como 192.168.15.1?

```
egrep "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
```

3. Como o comando grep pode ser usado para listar o conteúdo do arquivo /etc/services, descartando todos os comentários (linhas começando com #)?

```
grep -v ^# /etc/services
```

4. O arquivo domains.txt contém uma lista de nomes de domínio, um por linha. Como o comando egrep seria usado para listar apenas domínios .org or .com?

```
egrep ".org$|.com$" domains.txt
```

Respostas aos Exercícios Exploratórios

1. A partir do diretório atual, como o comando `find` usaria uma expressão regular estendida para pesquisar todos os arquivos que não contêm um sufixo de arquivo padrão (nomes de arquivo que não terminam em `.txt` ou `.c`, por exemplo)?

```
find . -type f -regextype egrep -not -regex '.*\.[[:alnum:]]{1,}\$'
```

2. O comando `less` é o paginador padrão para exibir arquivos de texto longos no ambiente shell. Ao digitar `/`, uma expressão regular pode ser inserida no prompt de pesquisa para pular para a primeira correspondência pertinente. Para permanecer na posição atual do documento e destacar apenas as correspondências pertinentes, qual combinação de teclas deve ser inserida no prompt de pesquisa?

Pressionando `Ctrl + K` antes de inserir a expressão de pesquisa.

3. Em `less`, como seria possível filtrar a saída para que apenas as linhas que correspondem a uma expressão regular sejam exibidas?

Pressionando `&` e inserindo a expressão de pesquisa.



103.7 Lição 2

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.7 Pesquisar em arquivos de texto usando expressões regulares
Lição:	2 de 2

Introdução

O streaming de dados por meio de uma cadeia de comandos canalizados permite a aplicação de filtros compostos com base em expressões regulares. As expressões regulares são uma técnica importante usada não apenas na administração de sistemas, mas também na *mineração de dados* e áreas relacionadas. Dois comandos são especialmente adequados para manipular arquivos e texto usando expressões regulares: `grep` e `sed`. `grep` é um localizador de padrões e `sed` é um editor de fluxo. Eles são úteis por si sós, mas é no trabalho em conjunto com outros processos que se destacam de fato.

O localizador de padrões: grep

Um dos usos mais comuns do `grep` é facilitar a inspeção de arquivos longos, usando a expressão regular como filtro aplicado a cada linha. Pode ser usado para mostrar apenas as linhas que começam com um determinado termo. Por exemplo, o `grep` pode ser usado para investigar um arquivo de configuração para módulos do kernel, listando apenas as linhas de opções:

```
$ grep '^options' /etc/modprobe.d/alsa-base.conf
```

```
options snd-pcsp index=-2
options snd-usb-audio index=-2
options bt87x index=-2
options cx88_alsa index=-2
options snd-atiixp-modem index=-2
options snd-intel8x0m index=-2
options snd-via82xx-modem index=-2
```

O caractere pipe | pode ser empregado para redirecionar a saída de um comando diretamente para a entrada de grep. O exemplo a seguir usa uma expressão entre colchetes para selecionar linhas da saída de fdisk -l, começando com Disk /dev/sda ou Disk /dev/sdb:

```
# fdisk -l | grep '^Disk /dev/sd[a|b]'

Disk /dev/sda: 320.1 GB, 320072933376 bytes, 625142448 sectors
Disk /dev/sdb: 7998 MB, 7998537728 bytes, 15622144 sectors
```

A mera seleção de linhas com correspondências pode não ser apropriada para uma tarefa em particular, exigindo ajustes no comportamento de grep através de suas opções. Por exemplo, a opção -c ou --count diz ao grep para exibir quantas linhas têm correspondências:

```
# fdisk -l | grep '^Disk /dev/sd[a|b]' -c

2
```

A opção pode ser colocada antes ou depois da expressão regular. Outras opções importantes do grep são:

-c ou --count

Em vez de exibir os resultados da pesquisa, exibe apenas a contagem total de quantas vezes uma correspondência ocorre em qualquer arquivo.

-i ou --ignore-case

Torna a busca insensível a maiúsculas e minúsculas.

-f FILE ou --file=FILE

Indica um arquivo contendo a expressão regular a ser usada.

-n ou --line-number

Mostra o número da linha.

-v ou --invert-match

Seleciona todas as linhas, exceto aquelas que contêm correspondências.

-H ou --with-filename

Mostra também o nome do arquivo que contém a linha.

-z ou --null-data

Em vez de fazer com que o grep trate os fluxos de dados de entrada e saída como linhas separadas (usando *newline* por padrão), ele passa a encarar a entrada ou saída como uma sequência de linhas. Ao combinar a saída do comando `find` usando a opção `-print0` com o comando `grep`, a opção `-z` ou `--null-data` deve ser usada para processar o fluxo da mesma maneira.

Embora ativada por padrão quando vários caminhos de arquivo são fornecidos como entrada, a opção `-H` não é ativada para arquivos individuais. Isso pode ser importante em situações especiais, como quando `grep` é chamado diretamente por `find`, por exemplo:

```
$ find /usr/share/doc -type f -exec grep -i '3d modeling' "{}" \; | cut -c -100
artistic aspects of 3D modeling. Thus this might be the application you are
This major approach of 3D modeling has not been supported
oce is a C++ 3D modeling library. It can be used to develop CAD/CAM softwares, for
instance [FreeCad]
```

Neste exemplo, `find` lista todos os arquivos em `/usr/share/doc` e passa cada um deles para o `grep`, que por sua vez realiza uma busca sem distinção entre maiúsculas e minúsculas por `3d modeling` dentro do arquivo. O pipe para `cut` existe apenas para limitar o comprimento da saída a 100 colunas. Observe, entretanto, que não há como saber de qual arquivo as linhas vieram. Esse problema é resolvido adicionando `-H` ao `grep`:

```
$ find /usr/share/doc -type f -exec grep -i -H '3d modeling' "{}" \; | cut -c -100
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might
be the applicatio
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has
not been support
```

Agora é possível identificar os arquivos nos quais cada correspondência foi encontrada. Para tornar a listagem ainda mais informativa, linhas iniciais e finais podem ser adicionadas às linhas com correspondências:

```
$ find /usr/share/doc -type f -exec grep -i -H -1 '3d modeling' "{}" \; | cut -c
```

-100

```
/usr/share/doc/openscad/README.md-application Blender), OpenSCAD focuses on the CAD
aspects rather t
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might
be the applicatio
/usr/share/doc/openscad/README.md-looking for when you are planning to create 3D
models of machine p
/usr/share/doc/opencsg/doc/publications.html-3D graphics library for Constructive
Solid Geometry (CS
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has
not been support
/usr/share/doc/opencsg/doc/publications.html-by real-time computer graphics until
recently.
```

A opção `-1` instrui o `grep` a incluir uma linha antes e uma linha depois quando encontrar uma linha com uma correspondência. Essas linhas extras são chamadas de *linhas de contexto* e são identificadas na saída por um sinal de menos após o nome do arquivo. O mesmo resultado pode ser obtido com `-C 1` ou `--context=1`; outras quantidades de linhas de contexto podem ser indicadas.

Existem dois programas complementares a `grep`: `egrep` e `fgrep`. O programa `egrep` é equivalente ao comando `grep -E`, que incorpora recursos extras além das expressões regulares básicas. Por exemplo, com o `egrep` é possível usar recursos de expressões regulares estendidas, como as alternâncias:

```
$ find /usr/share/doc -type f -exec egrep -i -H -1 '3d (modeling|printing)' "{}" \;
| cut -c -100
/usr/share/doc/openscad/README.md-application Blender), OpenSCAD focuses on the CAD
aspects rather t
/usr/share/doc/openscad/README.md:artistic aspects of 3D modeling. Thus this might
be the applicatio
/usr/share/doc/openscad/README.md-looking for when you are planning to create 3D
models of machine p
/usr/share/doc/openscad/RELEASE_NOTES.md-* Support for using 3D-Mouse / Joystick /
Gamepad input dev
/usr/share/doc/openscad/RELEASE_NOTES.md-* 3D Printing support: Purchase from a
print service partne
/usr/share/doc/openscad/RELEASE_NOTES.md-* New export file formats: SVG, 3MF, AMF
/usr/share/doc/opencsg/doc/publications.html-3D graphics library for Constructive
Solid Geometry (CS
/usr/share/doc/opencsg/doc/publications.html:This major approach of 3D modeling has
not been support
/usr/share/doc/opencsg/doc/publications.html-by real-time computer graphics until
recently.
```

Neste exemplo, a expressão buscada corresponderá a `3D modeling` ou `3D printing`, sem distinção entre maiúsculas e minúsculas. Para exibir apenas as partes de um fluxo de texto que correspondem à expressão usada por `egrep`, use a opção `-o`.

O programa `fgrep` é equivalente a `grep -F`, ou seja, não analisa expressões regulares. Ele é útil em pesquisas simples, nas quais o objetivo é encontrar uma expressão literal. Portanto, caracteres especiais como o círculo e o ponto serão interpretados literalmente e não como seus significados em uma expressão regular.

O editor de fluxo: sed

O objetivo do programa `sed` é modificar dados baseados em texto de forma não interativa. Isso significa que toda a edição é feita por instruções predefinidas, não por uma digitação direta e arbitrária em um texto exibido na tela. Em termos modernos, o `sed` pode ser entendido como um analisador de modelo: dado um texto como entrada, ele coloca um conteúdo personalizado em posições predefinidas ou quando encontra uma correspondência para uma expressão regular.

O `sed` é adequado para texto transmitido por meio de encadeamentos (pipelines). Sua sintaxe básica é `sed -f SCRIPT`, quando as instruções de edição são armazenadas no arquivo `SCRIPT`, ou `sed -e COMANDOS` para executar `COMANDOS` diretamente da linha de comando. Se nem `-f` nem `-e` estiverem presentes, o `sed` usa o primeiro parâmetro que não seja uma opção como arquivo de script. Também é possível usar um arquivo como entrada apenas fornecendo seu caminho como um argumento para o `sed`.

As instruções do `sed` são compostas por um único caractere, possivelmente precedido por um endereço ou seguido por uma ou mais opções, e são aplicadas a uma linha de cada vez. Os endereços podem ser um único número de linha, uma expressão regular ou um intervalo de linhas. Por exemplo, a primeira linha de um fluxo de texto pode ser excluída com `1d`, onde `1` especifica a linha à qual o comando de exclusão `d` será aplicado. Para esclarecer o uso de `sed`, vejamos a saída do comando `factor `seq 12``, que retorna os fatores primos para os números de 1 a 12:

```
$ factor `seq 12'  
1:  
2: 2  
3: 3  
4: 2 2  
5: 5  
6: 2 3  
7: 7  
8: 2 2 2  
9: 3 3  
10: 2 5
```

```
11: 11
12: 2 2 3
```

A exclusão da primeira linha com o `sed` é realizada por `1d`:

```
$ factor `seq 12` | sed 1d
2: 2
3: 3
4: 2 2
5: 5
6: 2 3
7: 7
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

Especificamos um intervalo de linhas com uma vírgula de separação:

```
$ factor `seq 12` | sed 1,7d
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

Mais de uma instrução pode ser usada na mesma execução, separadas por ponto e vírgula. Nesse caso, no entanto, é importante colocá-las entre parênteses para que o ponto e vírgula não seja interpretado pelo shell:

```
$ factor `seq 12` | sed "1,7d;11d"
8: 2 2 2
9: 3 3
10: 2 5
12: 2 2 3
```

Neste exemplo, duas instruções de exclusão foram executadas, primeiro nas linhas que vão de 1 a 7 e depois na linha 11. Um endereço também pode ser uma expressão regular, portanto, apenas as linhas correspondentes serão afetadas pela instrução:

```
$ factor `seq 12` | sed "1d;/:.*2.*/d"
3: 3
5: 5
7: 7
9: 3 3
11: 11
```

A expressão regular `:.*2.*` corresponde a qualquer ocorrência do número 2 em qualquer lugar depois de dois pontos, provocando a exclusão das linhas correspondentes aos números que tenham 2 como fator. Com o `sed`, qualquer coisa colocada entre barras (/) é considerada uma expressão regular e, por padrão, todas as ER básicas são suportadas. Por exemplo, `sed -e "/^#/d" /etc/services` mostra o conteúdo do arquivo `/etc/services` sem as linhas que começam com `#` (linhas de comentário).

A instrução de exclusão `d` é apenas uma das muitas instruções de edição fornecidas pelo `sed`. Em vez de excluir uma linha, o `sed` pode substituí-la por um texto determinado:

```
$ factor `seq 12` | sed "1d;/:.*2.*/c REMOVED"
REMOVED
3: 3
REMOVED
5: 5
REMOVED
7: 7
REMOVED
9: 3 3
REMOVED
11: 11
REMOVED
```

A instrução `c` `REMOVED` simplesmente substitui uma linha pelo texto `REMOVED`. No caso do exemplo, cada linha com uma substring correspondente à expressão regular `:.*2.*` é afetada pela instrução `c` `REMOVED`. A instrução `a` `TEXT` copia o texto indicado por `TEXT` para uma nova linha após a linha com uma correspondência. A instrução `r` `FILE` faz o mesmo, mas copia o conteúdo do arquivo indicado por `FILE`. A instrução `w` faz o oposto de `r`, ou seja, a linha será anexada ao arquivo indicado.

De longe, a instrução mais usada do `sed` é `s/FIND/REPLACE/`, que é usada para substituir uma correspondência da expressão regular `FIND` pelo texto indicado por `REPLACE`. Por exemplo, a instrução `s/hda/sda/` substitui uma substring que corresponde à ER literal `hda` por `sda`. Apenas a primeira correspondência encontrada na linha será substituída, a menos que o sinalizador `g` seja colocado após a instrução, como em `s/hda/sda/g`.

Um estudo de caso mais realístico ajudará a ilustrar os recursos do `sed`. Suponha que uma clínica médica queira enviar mensagens de texto a seus clientes, lembrando-os de suas consultas agendadas para o dia seguinte. Um cenário de implementação típico depende de um serviço profissional de mensagens instantâneas, que fornece uma API para acessar o sistema responsável pela entrega das mensagens. Essas mensagens geralmente são originadas do mesmo sistema que executa o aplicativo de controle de agendamentos do cliente, acionadas por um horário específico do dia ou algum outro evento. Nessa situação hipotética, o aplicativo poderia gerar um arquivo chamado `appointments.csv` contendo dados tabulados com todas as consultas agendadas para o dia seguinte, que seria usado pelo `sed` para construir as mensagens de texto a partir de um arquivo de modelo chamado `template.txt`. Os arquivos CSV são uma forma padrão de exportar dados de agendamentos de banco de dados. Portanto, os dados padrão dos agendamentos poderiam ser fornecidos da seguinte forma:

```
$ cat appointments.csv
"NAME", "TIME", "PHONE"
"Carol", "11am", "55557777"
"Dave", "2pm", "33334444"
```

A primeira linha contém os rótulos de cada coluna, que serão usados para preencher as tags dentro do arquivo de modelo:

```
$ cat template.txt
Hey <NAME>, don't forget your appointment tomorrow at <TIME>.
```

Os sinais de menor que `<` e maior que `>` foram postos em torno dos rótulos apenas para ajudar a identificá-los como tags. O script Bash a seguir analisa todos os agendamentos da fila usando `template.txt` como modelo de mensagem:

```
#!/bin/bash

TEMPLATE='cat template.txt'
TAGS=(`sed -ne '1s/^"/;1s//,"/\n/g;1s/"$/p' appointments.csv`)
mapfile -t -s 1 ROWS < appointments.csv
for (( r = 0; r < ${#ROWS[*]}; r++ ))
do
  MSG=$TEMPLATE
  VALS=(`sed -e 's/^"/;s//,"/\n/g;s/"$/'' <<<${ROWS[$r]}`')
  for (( c = 0; c < ${#TAGS[*]}; c++ ))
  do
    MSG=`sed -e "s/<${TAGS[$c]}>/${VALS[$c]}/g" <<<"$MSG"'
```

```

done
echo curl --data message=\"$MSG\" --data phone=\"${VALS[2]}\""
https://mysmsprovider/api
done

```

Um script de produção real também cuidaria da autenticação, verificação de erros e registro, mas o exemplo mostra uma funcionalidade básica para começar. As primeiras instruções executadas pelo `sed` são aplicadas apenas à primeira linha – o endereço 1 em `1s/^"/;1s/"//\n/g;1s://"p` – para remover as aspas iniciais e finais – `1s/^"/` e `1s://"p` – e substituir os separadores de campo por um caractere de nova linha: `1s://"//\n/g`. Apenas a primeira linha é necessária para carregar os nomes das colunas, de forma que as linhas não correspondentes serão suprimidas pela opção `-n`, exigindo que o sinalizador `p` seja colocado após o último comando `sed` para imprimir a linha correspondente. As tags são então armazenadas na variável `TAGS` como uma matriz Bash. Outra variável da matriz Bash é criada pelo comando `mapfile` para armazenar as linhas contendo os agendamentos na variável da matriz `ROWS`.

Um loop `for` é empregado para processar cada linha de agendamento encontrada em `ROWS`. Em seguida, as aspas e separadores no agendamento – o agendamento está na variável `${ROWS[$r]}` usada como uma *here string* – são substituídos por `sed`, como no caso dos comandos usados para carregar as tags. Os valores separados do agendamento são então armazenados na variável de matriz `VALS`, onde os subscritos de matriz 0, 1 e 2 correspondem aos valores de NAME, TIME e PHONE.

Finalmente, um loop `for` aninhado percorre a matriz `TAGS` e substitui cada tag encontrada no modelo por seu valor correspondente em `VALS`. A variável `MSG` contém uma cópia do modelo gerado, atualizado pelo comando de substituição `s/<${TAGS[$c]}>/${VALS[$c]}/g` em cada loop que passa por `TAGS`.

Isso resulta em uma mensagem semelhante a: "Hey Carol, don't forget your appointment tomorrow at 11am." (em português: "Olá, Carol. Não se esqueça de sua consulta amanhã às 11h."). A mensagem gerada pode então ser enviada como um parâmetro através de uma solicitação HTTP com `curl`, em forma de email ou qualquer outro método semelhante.

Combinando grep e sed

Os comandos `grep` e `sed` podem ser usados juntos quando uma mineração de texto mais complexa é necessária. Como administrador do sistema, você pode querer inspecionar todas as tentativas de login em um servidor, por exemplo. O arquivo `/var/log/wtmp` registra todos os logins e logouts, ao passo que `/var/log/btmp` registra as tentativas de login falhadas. Eles são escritos em um formato binário, que pode ser lido, respectivamente, pelos comandos `last` e `lastb`. A saída de `lastb` mostra não apenas o nome de usuário empregado na tentativa de login incorreta, mas também seu endereço IP:

```
# lastb -d -a -n 10 --time-format notime
user      ssh:notty      (00:00)      81.161.63.251
nrostagn ssh:notty      (00:00)      vmd60532.contaboserver.net
pi        ssh:notty      (00:00)      132.red-88-20-39.staticip.rima-tde.net
pi        ssh:notty      (00:00)      132.red-88-20-39.staticip.rima-tde.net
pi        ssh:notty      (00:00)      46.6.11.56
pi        ssh:notty      (00:00)      46.6.11.56
nps       ssh:notty      (00:00)      vmd60532.contaboserver.net
narmadan ssh:notty      (00:00)      vmd60532.contaboserver.net
nominati ssh:notty      (00:00)      vmd60532.contaboserver.net
nominati ssh:notty      (00:00)      vmd60532.contaboserver.net
```

A opção `-d` traduz o número IP para o nome de host correspondente. O nome do host pode fornecer pistas sobre o provedor de internet ou o serviço de hospedagem usado para realizar essas tentativas de login incorretas. A opção `-a` põe o nome do host na última coluna, o que facilita a filtragem a ser aplicada. A opção `--time-format notime` suprime a hora em que ocorreu a tentativa de login. O comando `lastb` pode levar algum tempo para ser concluído caso tenha havido muitas tentativas de login incorretas, por isso limitamos a saída a dez entradas com a opção `-n 10`. Nem todos os IPs remotos têm um nome de host associado, portanto o DNS reverso não se aplica e eles podem ser dispensados. Embora seja possível escrever uma expressão regular que corresponda ao formato esperado para um nome de host no final da linha, provavelmente é mais simples escrever uma que corresponda a uma letra do alfabeto ou a um único dígito no final da linha. O exemplo a seguir mostra como o comando `grep` pega a listagem em sua entrada padrão e remove as linhas sem nomes de host:

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | head -n 10
nvidia    ssh:notty      (00:00)      vmd60532.contaboserver.net
n_tonson  ssh:notty      (00:00)      vmd60532.contaboserver.net
nrostagn  ssh:notty      (00:00)      vmd60532.contaboserver.net
pi        ssh:notty      (00:00)      132.red-88-20-39.staticip.rima-tde.net
pi        ssh:notty      (00:00)      132.red-88-20-39.staticip.rima-tde.net
nps       ssh:notty      (00:00)      vmd60532.contaboserver.net
narmadan ssh:notty      (00:00)      vmd60532.contaboserver.net
nominati ssh:notty      (00:00)      vmd60532.contaboserver.net
nominati ssh:notty      (00:00)      vmd60532.contaboserver.net
nominati ssh:notty      (00:00)      vmd60532.contaboserver.net
```

A opção `-v` do comando `grep` exibe apenas as linhas que não correspondem à expressão regular fornecida. Uma expressão regular que corresponda a qualquer linha terminada por um número (isto é, `[0-9]$`) captura apenas as entradas sem um nome de host. Portanto, `grep -v '[0-9]$'` mostra somente as linhas que terminam com um nome de host. A filtragem da saída pode ser ainda mais refinada, mantendo-se apenas o nome de domínio e removendo as outras partes de cada

linha. Para isso, usamos o comando `sed` com um comando de substituição para trocar a linha inteira por um agrupamento referente ao nome de domínio contido nela:

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | sed -e 's/.* \(.*\)$/\1/' | head -n 10
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
132.red-88-20-39.staticip.rima-tde.net
132.red-88-20-39.staticip.rima-tde.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
vmd60532.contaboserver.net
```

O parêntese escapado em `.* \(.*\)$` diz ao `sed` para lembrar aquela parte da linha, ou seja, a parte entre o último caractere de espaço e o final da linha. No exemplo, esta parte é referenciada com `\1` e usada para substituir a linha inteira. Parece claro que a maioria dos hosts remotos tenta fazer o login mais de uma vez; assim, o mesmo nome de domínio se repete. Para suprimir as entradas repetidas, elas precisam primeiro ser classificadas (com o comando `sort`) e depois passadas para o comando `uniq`:

```
# lastb -d -a --time-format notime | grep -v '[0-9]$' | sed -e 's/.* \(.*\)$/\1/' | sort | uniq | head -n 10
116-25-254-113-on-nets.com
132.red-88-20-39.staticip.rima-tde.net
145-40-33-205.power-speed.at
tor.laquadrature.net
tor.momx.site
ua-83-226-233-154.bbcust.telenor.se
vmd38161.contaboserver.net
vmd60532.contaboserver.net
vmi488063.contaboserver.net
vmi515749.contaboserver.net
```

Este exemplo mostra como diferentes comandos podem ser combinados para produzir o resultado desejado. A lista de nomes de host pode então ser usada para escrever regras de firewall de bloqueio ou tomar outras medidas para garantir a segurança do servidor.

Exercícios Guiados

1. O comando `last` mostra uma lista dos últimos usuários logados, incluindo seus IPs de origem. Como o comando `egrep` poderia ser usado para filtrar a saída de `last`, mostrando apenas as ocorrências de um endereço IPv4, descartando quaisquer informações adicionais na linha correspondente?

2. Qual opção deve ser dada ao `grep` para filtrar corretamente a saída gerada pelo comando `find` executado com a opção `-print0`?

3. O comando `uptime -s` mostra a última data em que o sistema foi ligado, como em `2019-08-05 20:13:22`. Qual será o resultado do comando `uptime -s | sed -e 's/(.*) (.*)/\1/'`?

4. Qual opção deve ser dada ao `grep` para que ele conte as linhas correspondentes aos termos de pesquisa em vez de exibi-las?

Exercícios Exploratórios

1. A estrutura básica de um arquivo HTML começa com os elementos `html`, `head` e `body`, como por exemplo:

```
<html>
<head>
  <title>News Site</title>
</head>
<body>
  <h1>Headline</h1>
  <p>Information of interest.</p>
</body>
</html>
```

Descreva como seria possível usar endereços no `sed` para exibir apenas o elemento `body` e seu conteúdo.

2. Qual expressão de `sed` poderia remover todas as tags de um documento HTML, mantendo apenas o texto?

3. Os arquivos com extensão `.ovpn` são muito populares para configurar clientes VPN, pois contêm não apenas as configurações, mas também o conteúdo de chaves e certificados para o cliente. Essas chaves e certificados estão originalmente em arquivos separados e portanto precisam ser copiados para o arquivo `.ovpn`. Dado o seguinte trecho de um modelo `.ovpn`:

```
client
dev tun
remote 192.168.1.155 1194
<ca>
ca.crt
</ca>
<cert>
client.crt
</cert>
<key>
client.key
</key>
<tls-auth>
```

```
ta.key  
</tls-auth>
```

Supondo-se que os arquivos `ca.crt`, `client.crt`, `client.key` e `ta.key` estejam no diretório atual, como a configuração do modelo seria modificada pelo `sed` de forma a substituir cada nome de arquivo pelo seu conteúdo?

Resumo

Esta lição trata dos dois comandos mais importantes do Linux relacionados a expressões regulares: `grep` e `sed`. Os scripts e comandos compostos contam com `grep` e `sed` para realizar uma ampla gama de tarefas de filtragem e análise de texto. A lição inclui as seguintes etapas:

- Como usar o `grep` e suas variações, como `egrep` e `fgrep`.
- Como usar o `sed` e suas instruções internas para manipular texto.
- Exemplos de aplicações de expressões regulares usando `grep` e `sed`.

Respostas aos Exercícios Guiados

- O comando `last` mostra uma lista dos últimos usuários logados, incluindo seus IPs de origem. Como o comando `egrep` poderia ser usado para filtrar a saída de `last`, mostrando apenas as ocorrências de um endereço IPv4, descartando quaisquer informações adicionais na linha correspondente?

```
last -i | egrep -o '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
```

- Qual opção deve ser dada ao `grep` para filtrar corretamente a saída gerada pelo comando `find` executado com a opção `-print0`?

A opção `-z` or `--null-data`, como em `find . -print0 | grep -z expression`.

- O comando `uptime -s` mostra a última data em que o sistema foi ligado, como em `2019-08-05 20:13:22`. Qual será o resultado do comando `uptime -s | sed -e 's/(.*) (.*)/\1/'`?

Ocorrerá um erro. Por padrão, os parênteses devem ser escapados para ser possível usar agrupamentos no `sed`.

- Qual opção deve ser dada ao `grep` para que ele conte as linhas correspondentes aos termos de pesquisa em vez de exibi-las?

A opção `-c`.

Respostas aos Exercícios Exploratórios

1. A estrutura básica de um arquivo HTML começa com os elementos `html`, `head` e `body`, como por exemplo:

```
<html>
<head>
  <title>News Site</title>
</head>
<body>
  <h1>Headline</h1>
  <p>Information of interest.</p>
</body>
</html>
```

Descreva como seria possível usar endereços no `sed` para exibir apenas o elemento `body` e seu conteúdo.

Para mostrar apenas `body`, os endereços deveriam ser `/<body>/, /<\body>/`, como em `sed -n -e '/<body>/, /<\body>/p'`. A opção `-n` é dada ao `sed` para que ele não imprima linhas por padrão, por isso o comando `p` está no final da expressão de `sed` para imprimir as linhas correspondentes aos termos da pesquisa.

2. Qual expressão de `sed` poderia remover todas as tags de um documento HTML, mantendo apenas o texto?

A expressão do `sed` `s/<[^>]*>//g` substitui qualquer conteúdo entre `<>` por uma string vazia.

3. Os arquivos com extensão `.ovpn` são muito populares para configurar clientes VPN, pois contêm não apenas as configurações, mas também o conteúdo de chaves e certificados para o cliente. Essas chaves e certificados estão originalmente em arquivos separados e portanto precisam ser copiados para o arquivo `.ovpn`. Dado o seguinte trecho de um modelo `.ovpn`:

```
client
dev tun
remote 192.168.1.155 1194
<ca>
ca.crt
</ca>
<cert>
client.crt
</cert>
```

```
<key>
client.key
</key>
<tls-auth>
ta.key
</tls-auth>
```

Supondo-se que os arquivos ca.crt, client.crt, client.key e ta.key estejam no diretório atual, como a configuração do modelo seria modificada pelo sed de forma a substituir cada nome de arquivo pelo seu conteúdo?

O comando

```
sed -r -e 's/(^[^.]*).(\crt|key)$/cat \1.\2/e' < client.template > client.ovpn
```

substitui qualquer linha que termine em .crt ou .key pelo conteúdo de um arquivo cujo nome seja igual ao da linha. A opção -r diz ao sed para usar expressões regulares estendidas, ao passo que o e no final da expressão diz ao sed para substituir as correspondências pela saída do comando cat \1.\2. Os agrupamentos \1 e \2 correspondem ao nome de arquivo e extensão encontrados na correspondência.



103.8 Edição básica de arquivos com o vi

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.8

Peso

3

Áreas chave de conhecimento

- Navegar pelo documento usando o vi.
- Usar os modos básicos do vi.
- Inserir, editar, deletar, copiar e encontrar texto.
- Noções de Emacs, nano e vim.
- Configurar o editor padrão.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- vi
- /, ?
- h, j, k, l
- i, o, a
- d, p, y, dd, yy
- ZZ, :w!, :q!
- EDITOR



103.8 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.8 Edição básica de arquivos
Lição:	1 de 1

Introdução

Na maioria das distribuições Linux, o `vi` – abreviatura de “visual” – já vem pré-instalado e é o editor padrão do ambiente shell. O `vi` é um editor de texto interativo, que mostra o conteúdo do arquivo na tela à medida que ele é editado. Como tal, permite ao usuário mover-se e fazer modificações em qualquer parte do documento. No entanto, ao contrário dos editores visuais de uma área de trabalho gráfica, o editor `vi` é um aplicativo do shell com atalhos de teclado para todas as tarefas de edição.

Uma alternativa ao `vi`, chamada `vim` (*vi improved*), é algumas vezes usada como um substituto moderno do `vi`. Dentre outras melhorias, o `vim` oferece suporte para realce de sintaxe, desfazer/refazer multinível e edição de documentos múltiplos. Embora tenha mais recursos, o `vim` é totalmente compatível com as versões anteriores do `vi`, sendo ambos indistinguíveis para a maioria das tarefas.

A maneira padrão de iniciar o `vi` é fornecer a ele um caminho para um arquivo como parâmetro. Para pular diretamente para uma linha específica, o número dela deve ser informado com um sinal de mais, como em `vi +9 /etc/fstab` para abrir `/etc/fstab/` e posicionar o cursor na 9ª linha. Sem um número, o sinal de mais sozinho coloca o cursor na última linha.

A interface do `vi` é muito simples: todo o espaço disponível na janela do terminal é ocupado para

apresentar um arquivo, normalmente informado como um argumento de comando, ao usuário. As únicas dicas visuais são uma linha de rodapé mostrando a posição atual do cursor e um til ~ indicando onde o arquivo termina. Existem diferentes modos de execução para o vi nos quais o comportamento do programa muda. Os mais comuns são: *modo de inserção* e *modo normal*.

Modo de inserção

O modo de inserção é bem direto: o texto vai aparecendo na tela conforme é digitado no teclado. É o tipo de interação que a maioria dos usuários espera de um editor de texto, mas não é assim que o vi apresenta um documento logo de cara. Para entrar no modo de inserção, o usuário precisa executar um comando de inserção no modo normal. A tecla `Esc` conclui o modo de inserção e retorna ao modo normal, o modo padrão do vi.

Se quiser saber mais sobre os outros modos de execução, abra o vi e digite:

NOTE

```
:help vim-modes-intro
```

Modo normal

O modo normal — também conhecido como modo de comando — é como o vi inicia por padrão. Neste modo, as teclas do teclado são associadas a comandos de navegação e a tarefas de manipulação de texto. A maioria dos comandos neste modo são teclas únicas. Eis algumas das teclas e suas funções no modo normal:

0, \$

Ir para o início e o fim da linha.

1G, G

Ir para o início e o fim do documento.

(,)

Ir para o início e o fim da frase.

{, }

Ir para o início e o fim do parágrafo.

w, W

Pular palavra e pular palavra incluindo a pontuação.

h, j, k, l

Pra esquerda, pra baixo, pra cima, pra direita.

e or E

Ir para o fim da palavra atual.

/, ?

Pesquisa para a frente e para trás.

i, I

Entrar no modo de inserção antes da posição atual do cursor e no início da linha atual.

a, A

Entrar no modo de inserção após a posição atual do cursor e no final da linha atual.

o, O

Adicionar uma nova linha e entrar no modo de inserção na próxima linha ou na linha anterior.

s, S

Apagar o caractere sob o cursor ou a linha inteira e entrar no modo de inserção.

c

Alterar o(s) caractere(s) sob o cursor.

r

Substituir o caractere sob o cursor.

x

Excluir os caracteres selecionados ou o caractere sob o cursor.

v, V

Iniciar uma nova seleção com o caractere atual ou a linha inteira.

y, yy

Copia (arranca) o(s) caractere(es) ou a linha inteira.

p, P

Colar o conteúdo copiado, antes ou depois da posição atual.

U

Desfazer a última ação.

Ctrl-R

Refazer a última ação.

ZZ

Fechar e salvar.

ZQ

Fechar e não salvar.

Se precedido por um número, o comando será executado o mesmo número de vezes. Por exemplo, pressione `3yy` para copiar a linha atual mais as duas seguintes, pressione `d5w` para deletar a palavra atual e as 4 palavras seguintes, e assim por diante.

A maioria das tarefas de edição são combinações de vários comandos. Por exemplo, a sequência de teclas `vey` é usada para copiar uma seleção começando na posição atual até o final da palavra atual. A repetição de comandos também pode ser usada em combinações, então `v3ey` copiaria uma seleção começando na posição atual até o final da terceira palavra a partir de lá.

O `vi` pode organizar o texto copiado em registros, permitindo manter conteúdos distintos ao mesmo tempo. Um registro é especificado por um caractere precedido por `"` e, uma vez criado, é mantido até o final da sessão atual. A seqüência de teclas `"ly` cria um registro contendo a seleção atual, que estará acessível através da tecla `l`. Então, o registro `l` pode ser colado com `"lp`.

Também existe uma maneira de definir marcas personalizadas em posições arbitrárias ao longo do texto, sendo assim mais fácil alternar rapidamente entre elas. As marcas são criadas pressionando a tecla `m` e, em seguida, uma tecla para endereçar a posição atual. Feito isso, o cursor voltará para a posição marcada quando `'` seguido pela tecla escolhida for pressionada.

Qualquer sequência de teclas pode ser gravada como uma macro para execução futura. Podemos gravar uma macro, por exemplo, para colocar um texto selecionado entre aspas duplas. Primeiro, uma string de texto é selecionada e a tecla `q` é pressionada, seguida por uma tecla de registro que será associada à macro, como `d`. A linha `recording @d` aparecerá na linha de rodapé, indicando que a gravação está ativada. Presume-se que já haja algum texto selecionado, então o primeiro comando é `x` para remover (e copiar automaticamente) o texto selecionado. A tecla `i` é pressionada para inserir duas aspas duplas na posição atual, e depois `Esc` para retornar ao modo normal. O último comando é `P`, para inserir novamente a seleção excluída antes da última aspa dupla. Pressionar `q` novamente encerrará a gravação. Agora, uma macro que consiste na sequência de teclas `x, i, "", Esc` e `P` será executada toda vez que as teclas `@d` forem pressionadas no modo normal, sendo `d` a tecla de registro

associada à macro.

No entanto, a macro estará disponível apenas durante a sessão atual. Para tornar as macros persistentes, elas devem ser armazenadas no arquivo de configuração. Como a maioria das distribuições modernas usa o `vim` como editor compatível com `vi`, o arquivo de configuração do usuário é `~/.vimrc`. Dentro de `~/.vimrc`, a linha `let @d = 'xi"" P'` define o registro `d` para a sequência de teclas entre aspas simples. O mesmo registro atribuído anteriormente a uma macro pode ser usado para colar sua sequência de teclas.

Comandos de dois pontos

O modo normal também suporta outro conjunto de comandos `vi`: os comandos de *dois pontos*. Os comandos de dois pontos, como o nome indica, são executados após pressionar a tecla de dois pontos `:` no modo normal. Os comandos de dois pontos permitem ao usuário realizar pesquisas, salvar, sair, executar comandos do shell, alterar as configurações do `vi`, etc. Para voltar ao modo normal, executamos o comando `:visual` ou pressionamos a tecla Enter sem qualquer comando. Indicamos a seguir alguns dos comandos de dois pontos mais comuns (a inicial não faz parte do comando):

`:s/REGEX/TEXT/g`

Substituir todas as ocorrências da expressão regular REGEX por TEXT na linha atual. Ele aceita a mesma sintaxe do comando `sed`, incluindo endereços.

`:!`

Rodar um comando do shell a seguir.

`:quit or :q`

Sair do programa.

`:quit! or :q!`

Sair do programa sem salvar.

`:wq`

Salvar e sair.

`:exit or :x or :e`

Salvar e sair, se necessário.

`:visual`

Voltar ao modo de navegação.

O programa `vi` padrão é capaz de realizar a maioria das tarefas de edição de texto, mas qualquer outro editor não-gráfico pode ser usado para editar arquivos de texto no ambiente shell.

TIP

Os usuários novatos podem ter dificuldade para memorizar todas as teclas de comando do `vi` de uma vez. As distribuições que adotam o `vim` também possuem o comando `vimtutor`, que usa o próprio `vim` para abrir um guia passo a passo das principais atividades. O arquivo é uma cópia editável que pode ser usada para praticar os comandos e se acostumar progressivamente com eles.

Editores alternativos

Os usuários não familiarizados com o `vi` podem ter dificuldade de adaptação a ele, pois seu funcionamento não é intuitivo. Uma alternativa mais simples é o `nano` do GNU, um pequeno editor de texto que oferece todos os recursos básicos de edição de texto como desfazer/refazer, realce de sintaxe, busca e substituição interativa, recuo automático, números de linha, completar palavras, bloqueio de arquivos, arquivos de backup e suporte à internacionalização. Ao contrário do `vi`, todas as teclas pressionadas são simplesmente inseridas no documento que está sendo editado. Os comandos no `nano` são dados usando a tecla `Ctrl` ou a tecla Meta (dependendo do sistema, Meta é `Alt` ou `Esc`).

Ctrl-**A**

Iniciar uma nova seleção. Também é possível criar uma seleção pressionando Shift e movendo o cursor.

Meta-**C**

Copiar a seleção atual.

Ctrl-**K**

Cortar a seleção atual.

Ctrl-**U**

Colar o conteúdo copiado.

Meta-**U**

Desfazer.

Meta-**E**

Refazer.

Ctrl-

Substituir o texto na seleção.

Ctrl-T

Iniciar uma sessão de verificação ortográfica para o documento ou seleção atual.

O Emacs é outro editor de texto muito popular para o ambiente de shell. Ao passo que o texto é inserido com a digitação simples, como no `nano`, a navegação é auxiliada por comandos do teclado, como no `vi`. O Emacs inclui muitos recursos que o tornam mais do que apenas um editor de texto. Também é um IDE (*ambiente de desenvolvimento integrado*) capaz de compilar, executar e testar programas. O Emacs pode ser configurado como cliente de email, notícias ou RSS, tornando-o um verdadeiro pacote de produtividade.

O próprio shell executa um editor de texto padrão, normalmente o `vi`, sempre que necessário. É o que acontece, por exemplo, quando `crontab -e` é executado para editar *cronjobs*. O Bash usa as variáveis de sessão `VISUAL` ou `EDITOR` para encontrar o editor de texto padrão para o ambiente shell. Por exemplo, o comando `export EDITOR=nano` define o `nano` como editor de texto padrão na sessão do shell atual. Para tornar essa mudança persistente entre as sessões, o comando deve ser incluído em `~/.bash_profile`.

Exercícios Guiados

- O `vi` é usado principalmente como editor de arquivos de configuração e código-fonte, onde a indentação ajuda a identificar seções de texto. Uma seleção pode ser recuada para a esquerda pressionando `<` e para a direita pressionando `>`. Quais teclas devem ser pressionadas no modo normal para recuar a seleção atual três passos para a esquerda?

- Uma linha inteira pode ser selecionada pressionando `V` no modo normal do `vi`. No entanto, o caractere de término da nova linha também será incluído. Quais teclas devem ser pressionadas no modo normal para selecionar a partir do caractere inicial até o caractere de nova linha, sem incluí-lo?

- Como o `vi` deve ser executado na linha de comando para abrir `~/.bash_profile` e pular direto para a última linha?

- Quais teclas devem ser pressionadas no modo normal do `vi` para excluir caracteres desde a posição atual do cursor até o caractere de ponto final seguinte?

Exercícios Exploratórios

- O `vim` permite selecionar blocos de texto com largura arbitrária, não apenas seções com linhas inteiras. Ao pressionar `Ctrl + V` no modo normal, uma seleção é feita movendo o cursor para cima, para baixo, para a esquerda e para a direita. Usando esse método, como excluir um bloco começando no primeiro caractere da linha atual, contendo as próximas oito colunas e cinco linhas de texto?

- Uma sessão do `vi` foi interrompida por uma falha de energia inesperada. Ao reabrir o arquivo, o `vi` pergunta ao usuário se deseja recuperar o arquivo de troca (uma cópia automática feita pelo `vi`). O que o usuário deve fazer para descartar o arquivo de troca?

- Em uma sessão do `vim`, uma linha foi previamente copiada para o registro `l`. Qual combinação de teclas gravaria uma macro no registro `a` para colar a linha do registro `l` imediatamente antes da linha atual?

Resumo

Esta lição trata do editor de texto padrão para o ambiente shell do Linux: o editor `vi`. Embora intimidante para o usuário novato, o `vi` possui recursos que o tornam uma boa escolha para a edição técnica e não técnica de texto. A lição segue as seguintes etapas:

- Uso básico e recursos úteis do `vi`.
- O que é o `vim`—o `vi` aprimorado—e outros editores alternativos.
- Como definir o editor de texto padrão para o ambiente do shell.

Os comandos e procedimentos abordados foram:

- O editor `vi` e sua versão aprimorada `vim`.
- Edição de texto básica no `vi`.
- Os editores alternativos `emacs` e `nano`.

Respostas aos Exercícios Guiados

- O `vi` é usado principalmente como editor de arquivos de configuração e código-fonte, onde a indentação ajuda a identificar seções de texto. Uma seleção pode ser recuada para a esquerda pressionando `<` e para a direita pressionando `>`. Quais teclas devem ser pressionadas no modo normal para recuar a seleção atual três passos para a esquerda?

As teclas `3<`, que indicam três passos para a esquerda.

- Uma linha inteira pode ser selecionada pressionando `V` no modo normal do `vi`. No entanto, o caractere de término da nova linha também será incluído. Quais teclas devem ser pressionadas no modo normal para selecionar a partir do caractere inicial até o caractere de nova linha, sem incluí-lo?

As teclas `0v$h`, que significam `0` (“pular para o início de uma linha”), `v` (“iniciar a seleção de caracteres”), `$` (“ir ao final da linha”) e `h` (“voltar uma posição”).

- Como o `vi` deve ser executado na linha de comando para abrir `~/.bash_profile` e pular direto para a última linha?

O comando `vi + ~/.bash_profile` abre o arquivo e posiciona o cursor na última linha.

- Quais teclas devem ser pressionadas no modo normal do `vi` para excluir caracteres desde a posição atual do cursor até o caractere de ponto final seguinte?

As teclas `dt.`, que significam `d` (“iniciar a exclusão”), `t` (“pular para o próximo caractere”) and `.` (caractere de ponto final).

Respostas aos Exercícios Exploratórios

- O `vim` permite selecionar blocos de texto com largura arbitrária, não apenas seções com linhas inteiras. Ao pressionar `Ctrl + V` no modo normal, uma seleção é feita movendo o cursor para cima, para baixo, para a esquerda e para a direita. Usando esse método, como excluir um bloco começando no primeiro caractere da linha atual, contendo as próximas oito colunas e cinco linhas de texto?

A combinação `0, Ctrl + V e 8l5jd` seleciona e remove o bloco correspondente.

- Uma sessão do `vi` foi interrompida por uma falha de energia inesperada. Ao reabrir o arquivo, o `vi` pergunta ao usuário se deseja recuperar o arquivo de troca (uma cópia automática feita pelo `vi`). O que o usuário deve fazer para descartar o arquivo de troca?

Pressionar `d` quando solicitado pelo `vi`.

- Em uma sessão do `vim`, uma linha foi previamente copiada para o registro `l`. Qual combinação de teclas gravaria uma macro no registro `a` para colar a linha do registro `l` imediatamente antes da linha atual?

A combinação `qa"lPq`, que significa `q` (“iniciar a gravação da macro”), `a` (“atribuir o registro `a` à macro”), `"l` (“selecionar o texto no registro `l`”), `P` (“colar antes da linha atual”) e `q` (“encerrar a gravação da macro”).



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
```

serves 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

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 `fdisk`. 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 pertence 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 exclirmos 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

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, hifen, 4 dígitos, hifen, 4 dígitos, hifen, 4 dígitos, hifen, 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-29DFFEC36C0`.

-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`. 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 `resizelpart`. Como se trata da primeira partição, podemos pressupor que ela começa em zero e termina em 241 MB. Assim, o comando seria `resizelpart 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 3^a 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` pressupõe 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 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 BEHAVIOR` 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,avail,pcent`. Assim, a resposta é:

```
$ df -t ext4 --output=source,target,itotal,avail,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 automaticamente 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 exclirmos 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 sistemas 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,blksize=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 multusuário (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 usariamnos:

```
# 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 `umount`. 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 permissões 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+rwx,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 dos 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 `rxr-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=rx,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	<code>rw-</code>	<code>rwx</code>
1	<code>rw-</code>	<code>rw-</code>
2	<code>r--</code>	<code>r-x</code>
3	<code>r--</code>	<code>r--</code>

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 vírgula 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 significativo (*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!



104.6 Criar e alterar links simbólicos e hardlinks

Referência ao LPI objectivo

LPIC-1 version 5.0, Exam 101, Objective 104.6

Peso

2

Áreas chave de conhecimento

- Criar links.
- Identificar links simbólicos e/ou hardlinks.
- Copiar arquivos versus criar links de arquivos.
- Usar links para dar suporte a tarefas de administração do sistema.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `ln`
- `ls`



104.6 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	104 Dispositivos, sistemas de arquivos do Linux, hierarquia padrão de sistemas de arquivos
Objetivo:	104.6 Criar e alterar links físicos e simbólicos
Lição:	1 de 1

Introdução

No Linux, alguns arquivos recebem um tratamento especial, seja devido ao local em que estão armazenados, como os arquivos temporários, ou à maneira como interagem com o sistema de arquivos, como os links. Nesta lição, aprenderemos o que são os links e como gerenciá-los.

Compreendendo os links

Já dissemos que, no Linux, tudo é tratado como um arquivo. Mas existe um tipo *especial* de arquivo chamado *link*. Há dois tipos de links em um sistema Linux:

Links simbólicos

Também chamados de *soft links*, eles apontam para o caminho de outro arquivo. Se excluirmos o arquivo para o qual o link aponta (chamado *destino*), o link ainda existirá, mas “para de funcionar”, pois passará a apontar para “nada”.

Links físicos

Um link físico é como como um segundo nome para o arquivo original. Eles *não* são duplicatas,

mas sim uma entrada adicional no sistema de arquivos que aponta para o mesmo local (inode) no disco.

TIP Um *inode* é uma estrutura de dados que armazena os atributos de um objeto (como um arquivo ou diretório) em um sistema de arquivos. Entre esses atributos estão as permissões, proprietário e os blocos do disco nos quais estão armazenados os dados referentes àquele objeto. São semelhantes a um verbete em um índice; por isso o nome, que vem de “index node”.

Trabalhando com links físicos

Criando links físicos

O comando para criar um link físico no Linux é `ln`. A sintaxe básica é:

```
$ ln TARGET LINK_NAME
```

O `TARGET` já deve existir (trata-se do arquivo para o qual o link apontará) e, se o destino não estiver no diretório atual ou se você deseja criar o link em outro lugar, é *obrigatório* especificar o caminho completo para ele. Por exemplo, o comando

```
$ ln target.txt /home/carol/Documents/hardlink
```

Cria um arquivo chamado `hardlink` no diretório `/home/carol/Documents/`, vinculado ao arquivo `target.txt` no diretório atual.

Se deixarmos de fora o último parâmetro (`LINK_NAME`), será criado um link com o mesmo nome do destino no diretório atual.

Gerenciando os links físicos

Os links físicos são entradas no sistema de arquivos que têm nomes diferentes, mas que apontam para os mesmos dados no disco. Todos esses nomes são equivalentes e podem ser usados para se referir a um arquivo. Se você alterar o conteúdo de um dos nomes, o conteúdo de todos os outros nomes que apontam para aquele arquivo será alterado, já que todos apontam para os mesmos dados. Se você excluir um dos nomes, os outros nomes continuarão a funcionar.

Isso acontece porque quando você “exclui” um arquivo, os dados não são realmente apagados do disco. O sistema simplesmente exclui a entrada na tabela do sistema de arquivos apontando para o inode correspondente aos dados no disco. Mas se houver uma segunda entrada apontando para o

mesmo inode, ainda será possível acessar os dados. É como se fossem duas estradas convergindo no mesmo ponto. Mesmo se bloquearmos ou redirecionarmos uma das estradas, ainda será possível chegar ao destino usando a outra.

Para conferir, podemos usar o parâmetro `-i` de `ls`. Considere o seguinte conteúdo de um diretório:

```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

O número anterior às permissões é o número inode. Percebeu que tanto o arquivo `hardlink` quanto o arquivo `target.txt` têm o mesmo número (3806696)? Isso ocorre porque um é o link físico do outro.

Mas qual é o original e qual é o link? Não dá pra dizer, já que, na prática, eles são a mesma coisa.

Observe que todo link físico que aponta para um arquivo aumenta a *contagem de links* do arquivo. Este é o número logo após as permissões na saída de `ls -l`. Por padrão, todo arquivo tem uma contagem de links de 1 (os diretórios têm uma contagem de 2), e cada link físico apontando para ele aumenta a contagem em um. Por isso a contagem de links é 2 nos arquivos da lista acima.

Ao contrário dos links simbólicos, só é possível criar links físicos para arquivos, e tanto o link quanto o destino devem residir no mesmo sistema de arquivos.

Movendo e removendo links físicos

Como os links físicos são tratados como arquivos regulares, eles podem ser excluídos com `rm` e renomeados ou movidos no sistema de arquivos com `mv`. E como um link rígido aponta para o mesmo inode do destino, ele pode ser movido livremente, sem medo de “quebrar” o link.

Links simbólicos

Criando links simbólicos

O comando usado para criar um link simbólico também é `ln`, mas com o parâmetro `-s` adicionado. Assim:

```
$ ln -s target.txt /home/carol/Documents/softlink
```

Criamos assim um arquivo chamado `softlink` no diretório `/home/carol/Documents/`, apontando

para o arquivo `target.txt` no diretório atual.

Como no caso dos links físicos, é possível omitir o nome do link para criar um link com o mesmo nome do destino no diretório atual.

Gerenciando links simbólicos

Os links simbólicos apontam para outro caminho no sistema de arquivos. Podemos criar links simbólicos para arquivos e diretórios, mesmo em diferentes partições. É muito fácil identificar um link simbólico na saída de `ls`:

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
lrwxrwxrwx 1 carol carol    12 Jun  7 10:14 softlink -> target.txt
```

No exemplo acima, o primeiro caractere nas permissões para o arquivo `softlink` é `l`, indicando um link simbólico. Além disso, logo após o nome do arquivo, vemos o nome do destino para o qual o link aponta, o arquivo `target.txt`.

Note que nas listagens de arquivos e diretórios os links simbólicos sempre mostram as permissões `rwx` para o usuário, o grupo e outros, mas, na prática, as permissões de acesso para eles são as mesmas do destino.

Movendo e removendo links simbólicos

Como no caso dos links físicos, os links simbólicos podem ser removidos usando `rm` e movidos ou renomeados com `mv`. No entanto, deve-se tomar cuidado especial ao criá-los, para evitar “quebrar” o link se ele for movido de seu local original.

Ao criar links simbólicos, é preciso estar ciente de que, a menos que um caminho seja totalmente especificado, o local do destino será interpretado como sendo *relativo* ao local do link. Isso pode criar problemas caso o link ou o arquivo para o qual ele aponta seja movido.

É mais fácil entender isso com um exemplo. Digamos que temos um arquivo chamado `original.txt` no diretório atual e desejamos criar um link simbólico para ele chamado `softlink`. Poderíamos usar:

```
$ ln -s original.txt softlink
```

E aparentemente tudo correria bem. Vamos verificar com `ls`:

```
$ ls -lh
total 112K
-r--r--r-- 1 carol carol 110K Jun  7 10:13 original.txt
lrwxrwxrwx 1 carol carol    12 Jun  7 19:23 softlink -> original.txt
```

Observe como o link é elaborado: `softlink` aponta para (`→`) `original.txt`. Entretanto, vamos ver o que acontece se movermos o link para o diretório anterior e tentarmos exibir seu conteúdo com o comando `less`:

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

Como o caminho para `original.txt` não foi especificado, o sistema pressupõe que ele está no mesmo diretório que o link. Quando isso deixa de ser verdade, o link para de funcionar.

Para evitar isso, devemos sempre especificar o caminho completo para o destino ao criar o link:

```
$ ln -s /home/carol/Documents/original.txt softlink
```

Dessa forma, o link continuará a funcionar mesmo se for movido, porque aponta para a localização absoluta do destino. Verifique com `ls`:

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol   40 Jun  7 19:34 softlink ->
/home/carol/Documents/original.txt
```

Exercícios Guiados

1. Qual é o parâmetro de `chmod` no modo *simbólico* para ativar o sticky bit em um diretório?

2. Imagine que existe um arquivo chamado `document.txt` no diretório `/home/carol/Documents`. Com qual comando criariamos um link simbólico para ele chamado `text.txt` no diretório atual?

3. Explique a diferença entre um link físico para um arquivo e uma cópia desse arquivo.

Exercícios Exploratórios

- Imagine que dentro de um diretório você crie um arquivo chamado `recipes.txt`. Dentro deste diretório, você também criará um link físico para este arquivo, chamado `receitas.txt`, e um link simbólico (ou *soft*) para este chamado `rezepte.txt`.

```
$ touch recipes.txt
$ ln recipes.txt receitas.txt
$ ln -s recipes.txt rezepte.txt
```

O conteúdo do diretório deve aparecer assim:

```
$ ls -lhi
total 160K
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
5388837 lrwxrwxrwx 1 carol carol 12 jun 24 10:12 rezepte.txt -> receitas.txt
```

Lembre-se que, por ser um link físico, `receitas.txt` aponta para o mesmo inode que `recipes.txt`. O que aconteceria com o link `rezepte.txt` se o nome `receitas.txt` fosse excluído? Por quê?

- Imagine que você tem um pendrive conectado ao sistema e montado em `/media/youruser/FlashA`. Você deseja criar em seu diretório pessoal um link chamado `schematics.pdf` apontando para o arquivo `esquema.pdf` no diretório raiz do pendrive. Então, você digita o comando:

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

O que aconteceria? Por quê?

- Considere a seguinte saída de `ls -lah`:

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol 77K jun 17 17:25 document.txt
```

```
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png  
-rw-r--r-- 4 carol carol 77K jun 17 17:25 text.txt
```

- Quantos links apontam para o arquivo document.txt?

- Trata-se de links físicos ou simbólicos?

- Qual parâmetro você deveria passar para ls para ver qual inode é ocupado por cada arquivo?

4. Imagine que você tem, em seu diretório ~/Documents, um arquivo chamado clients.txt contendo alguns nomes de clientes e um diretório chamado somedir. Dentro dele, existe um arquivo diferente, também chamado clients.txt, contendo nomes diferentes. Para replicar essa estrutura, use os seguintes comandos:

```
$ cd ~/Documents  
$ echo "John, Michael, Bob" > clients.txt  
$ mkdir somedir  
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Em seguida você cria um link dentro de somedir chamado partners.txt apontando para esse arquivo, com os comandos:

```
$ cd somedir/  
$ ln -s clients.txt partners.txt
```

Assim, a estrutura do diretório é:

```
Documents  
|-- clients.txt  
'-- somedir  
    |-- clients.txt  
    '-- partners.txt -> clients.txt
```

Agora, você move partners.txt de somedir para ~/Documents, e lista seu conteúdo.

```
$ cd ~/Documents/  
$ mv somedir/partners.txt .  
$ less partners.txt
```

O link ainda funciona? Se sim, qual arquivo terá seu conteúdo listado? Por quê?

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt  
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quais são as permissões de acesso de `partners.txt`? Por quê?

Resumo

Nesta lição, você aprendeu:

- O que são links.
- A diferença entre os links *simbólicos* e *físicos*.
- Como criar links.
- Como movê-los, renomeá-los ou excluí-los.

Os seguintes comandos foram abordados nesta lição:

- `ln`: O comando “link”. Sozinho, este comando cria um link físico. Com a opção `-s`, podemos criar um link *simbólico* ou *soft link*. Lembre-se de que os links físicos só podem residir na mesma partição e sistema de arquivos, e que os links simbólicos podem atravessar partições e sistemas de arquivos (inclusive armazenamento conectado em rede)
- O parâmetro `-i` do `ls`, que permite visualizar o número inode de um arquivo.

Respostas aos Exercícios Guiados

1. Qual é o parâmetro de `chmod` no modo *simbólico* para ativar o sticky bit em um diretório?

O símbolo do sticky bit no modo simbólico é `t`. Como queremos habilitar (adicionar) essa permissão no diretório, o parâmetro deve ser `+t`.

2. Imagine que existe um arquivo chamado `document.txt` no diretório `/home/carol/Documents`. Com qual comando criariamos um link simbólico para ele chamado `text.txt` no diretório atual?

`ln -s` é o comando para criar um link simbólico. Como é necessário especificar o caminho completo do arquivo para o qual estamos criando o link, o comando seria:

```
$ ln -s /home/carol/Documents/document.txt text.txt
```

3. Explique a diferença entre um link físico para um arquivo e uma cópia desse arquivo.

Um link físico é apenas outro nome para um arquivo. Mesmo que pareça uma duplicata do arquivo original, para todos os efeitos, o link e o original são iguais, pois apontam para os mesmos dados no disco. As alterações feitas no conteúdo do link serão refletidas no original e vice-versa. Uma cópia é uma entidade completamente independente, ocupando um lugar diferente no disco. As alterações na cópia não serão refletidas no original e vice-versa.

Respostas aos Exercícios Exploratórios

- Imagine que dentro de um diretório você crie um arquivo chamado `recipes.txt`. Dentro deste diretório, você também criará um link físico para este arquivo, chamado `receitas.txt`, e um link simbólico (ou *soft*) para este chamado `rezepte.txt`.

```
$ touch recipes.txt
$ ln recipes.txt receitas.txt
$ ln -s recipes.txt rezepte.txt
```

O conteúdo do diretório deve aparecer assim:

```
$ ls -lhi
total 160K
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
5388837 lrwxrwxrwx 1 carol carol 12 jun 24 10:12 rezepte.txt -> recipes.txt
```

Lembre-se que, por ser um link físico, `receitas.txt` aponta para o mesmo inode que `recipes.txt`. O que aconteceria com o link `rezepte.txt` se o nome `receitas.txt` fosse excluído? Por quê?

O link simbólico `rezepte.txt` deixaria de funcionar, porque os links simbólicos apontam para nomes, e não inodes, e o nome `receitas.txt` não existe mais, mesmo que os dados ainda estejam no disco com o nome `recipes.txt`.

- Imagine que você tem um pendrive conectado ao sistema e montado em `/media/youruser/FlashA`. Você deseja criar em seu diretório pessoal um link chamado `schematics.pdf` apontando para o arquivo `esquema.pdf` no diretório raiz do pendrive. Então, você digita o comando:

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

O que aconteceria? Por quê?

O comando falharia. A mensagem de erro seria `Invalid cross-device link` (Link inválido entre dispositivos), esclarecendo o motivo: os links físicos não podem apontar para um destino em uma partição ou dispositivo diferente. A única maneira de criar um link como esse é usar um link simbólico ou *soft*, adicionando o parâmetro `-s` a `ln`.

3. Considere a seguinte saída de `ls -lah`:

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol 77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol 77K jun 17 17:25 text.txt
```

- Quantos links apontam para o arquivo `document.txt`?

Todos os arquivos começam com `1` na contagem de links. Como a contagem de links desse arquivo é `4`, existem três links apontando para ele.

- Trata-se de links físicos ou simbólicos?

São links físicos, já que os links simbólicos não aumentam a contagem de links de um arquivo.

- Qual parâmetro você deveria passar para `ls` para ver qual inode é ocupado por cada arquivo?

O parâmetro é `-i`. O inode será mostrado na primeira coluna da saída de `ls`, como abaixo:

```
$ ls -lahi
total 3,1M
5388773 drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
5245554 drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
5388840 -rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 document.txt
5388837 -rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 text.txt
```

4. Imagine que você tem, em seu diretório `~/Documents`, um arquivo chamado `clients.txt` contendo alguns nomes de clientes e um diretório chamado `somedir`. Dentro dele, existe um arquivo *diferente*, também chamado `clients.txt`, contendo nomes diferentes. Para replicar essa estrutura, use os seguintes comandos:

```
$ cd ~/Documents
```

```
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Em seguida você cria um link dentro de `somedir` chamado `partners.txt` apontando para esse arquivo, com os comandos:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Assim, a estrutura do diretório é:

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    '-- partners.txt -> clients.txt
```

Agora, você move `partners.txt` de `somedir` para `~/Documents`, e lista seu conteúdo.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

O link ainda funciona? Se sim, qual arquivo terá seu conteúdo listado? Por quê?

Isto é quase uma “pegadinha”, mas o link funcionará e o arquivo listado será o que está em `~/Documents`, contendo os nomes `John, Michael, Bob`.

Lembre-se de que, como não especificamos o caminho completo para o alvo `clients.txt` ao criar o link simbólico `partners.txt`, o local do destino será interpretado como sendo relativo ao local do link, que neste caso é o diretório atual.

Quando o link é movido de `~/Documents/somedir` para `~/Documents`, ele deveria parar de funcionar, já que o destino não estava mais no mesmo diretório do link. Porém, coincidentemente existe um arquivo chamado `clients.txt` em `~/Documents`, de forma que o link apontará para esse arquivo em vez do destino original dentro de `~/somedir`.

Para evitar isso, sempre especifique o caminho completo para o destino ao criar um link

simbólico.

5. Considere os arquivos a seguir:

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt  
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quais são as permissões de acesso de `partners.txt`? Por quê?

As permissões de acesso de `partners.txt` são `rw-r--r--`, já que os links sempre herdam as mesmas permissões de acesso do alvo.



104.7 Encontrar arquivos de sistema e conhecer sua localização correta

Referência ao LPI objectivo

LPIC-1 version 5.0, Exam 101, Objective 104.7

Peso

2

Áreas chave de conhecimento

- Entender a localização correta dos arquivos dentro do FHS.
- Encontrar arquivos e comandos em um sistema Linux.
- Conhecer a localização e a finalidade de arquivos e diretórios importantes definidos no FHS.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `find`
- `locate`
- `updatedb`
- `whereis`
- `which`
- `type`
- `/etc/updatedb.conf`



104.7 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	104 Dispositivos, sistemas de arquivos do Linux, hierarquia padrão de sistemas de arquivos
Objetivo:	104.7 Encontrar arquivos de sistema e colocar arquivos no local correto
Lição:	1 de 1

Introdução

As distribuições Linux vêm em todas as formas e tamanhos, mas uma coisa que quase todas têm em comum é o fato de seguirem o *Filesystem Hierarchy Standard* (FHS), que define um “layout padrão” para o sistema de arquivos. Isso simplifica muito a interoperação e administração do sistema. Nesta lição, você aprenderá mais sobre este padrão e como localizar arquivos em um sistema Linux.

O Filesystem Hierarchy Standard

O Filesystem Hierarchy Standard (FHS) é um projeto da Linux Foundation para padronizar a estrutura de diretórios e o conteúdo dos diretórios nos sistemas Linux. A conformidade com o padrão não é obrigatória, mas a maioria das distribuições o segue.

NOTE

Quem quiser saber mais sobre a organização dos sistemas de arquivos pode consultar a especificação do FHS 3.0 specification, disponível em vários formatos em: <http://refspecs.linuxfoundation.org/fhs.shtml>

De acordo com o padrão, a estrutura de diretórios básica é a seguinte:

/

Este é o diretório raiz, o primeiro da hierarquia. Todos os outros diretórios ficam dentro dele. Um sistema de arquivos muitas vezes é comparado a uma “árvore”; nesse caso, este seria o “tronco” ao qual todos os galhos se conectam.

/bin

Binários essenciais, disponível para todos os usuários.

/boot

Arquivos necessários ao processo de inicialização, incluindo o disco RAM inicial (initrd) e o próprio kernel do Linux.

/dev

Arquivos de dispositivos. Podem ser dispositivos físicos conectados ao sistema (por exemplo, /dev/sda seria o primeiro disco SCSI ou SATA) ou dispositivos virtuais fornecidos pelo kernel.

/etc

Arquivos de configuração específicos ao hospedeiro. Se necessário, os programas podem criar subdiretórios sob /etc para armazenar diversos arquivos de configuração.

/home

Cada usuário do sistema tem um diretório “inicial” para armazenar arquivos pessoais e preferências, a maioria deles localizados sob /home. Geralmente, o diretório home tem o mesmo nome do usuário, de modo que o usuário John teria seu diretório sob /home/john. As exceções são o superusuário (root), que tem um diretório separado (/root), e certos usuários do sistema.

/lib

Bibliotecas compartilhadas necessárias para inicializar o sistema operacional e executar os binários que estão em /bin e /sbin.

/media

Mídias removíveis montadas pelo usuário, como pendrives, leitores de CD e DVD-ROM, disquetes, cartões de memória e discos externos, são montadas aqui.

/mnt

Ponto de montagem para sistemas de arquivos montados temporariamente.

/opt

Pacotes de software opcionais.

/root

Diretório inicial do superusuário (root).

/run

Dados variáveis de tempo de execução.

/sbin

Binários do sistema.

/srv

Dados de serviços fornecidos pelo sistema. Por exemplo, as páginas fornecidas por um servidor web poderiam ser armazenadas em /srv/www.

/tmp

Arquivos temporários.

/usr

Dados de usuários de apenas leitura, incluindo dados utilizados por certos utilitários e aplicações secundárias.

/proc

Sistema de arquivos virtual contendo dados relacionados a processos em execução.

/var

Dados variáveis gravados durante a operação do sistema, incluindo fila de impressão, dados de log, caixas de entrada, arquivos temporários, cache do navegador etc.

Tenha em mente que alguns desses diretórios, como /etc, /usr e /var, contêm toda uma hierarquia de subdiretórios.

Arquivos Temporários

Os arquivos temporários são arquivos usados pelos programas para armazenar dados que serão necessários apenas por um curto período de tempo — por exemplo, dados de processos em execução, logs de falhas, arquivos de rascunho de um salvamento automático, arquivos intermediários usados durante uma conversão, arquivos de cache e assim por diante.

Localização dos arquivos temporários

A versão 3.0 do *Filesystem Hierarchy Standard* (FHS) define locais padrão para os arquivos

temporários nos sistemas Linux. Cada local tem uma finalidade e um comportamento diferentes, e é recomendável que os desenvolvedores sigam as convenções definidas pelo FHS ao gravar dados temporários no disco.

/tmp

De acordo com o FHS, não se deve pressupor que os arquivos escritos aqui serão preservados entre as invocações de um programa. A *recomendação* é que esse diretório seja limpo (todos os arquivos apagados) durante a inicialização do sistema, embora isso *não seja obrigatório*.

/var/tmp

Outro local para arquivos temporários, mas este *não deve ser limpo* durante a inicialização do sistema, ou seja, os arquivos armazenados aqui geralmente persistem entre as reinicializações.

/run

Este diretório contém arquivos variáveis de tempo de execução usados pelos processos ativos, como os arquivos identificadores de processo (.pid). Os programas que precisam de mais de um arquivo de tempo de execução podem criar subdiretórios aqui. Este local *deve ser limpo* durante a inicialização do sistema. Antigamente essa finalidade pertencia a /var/run e, em alguns sistemas, /var/run pode ser um link simbólico para /run.

Observe que nada impede o programa de criar arquivos temporários em outro local do sistema, mas é recomendável respeitar as convenções definidas pelo FHS.

Busca de arquivos

Para pesquisar por arquivos em um sistema Linux, podemos usar o comando `find`. Esta é uma ferramenta muito poderosa, rica em parâmetros configuráveis para modificar a saída exatamente de acordo com suas necessidades.

Para começar, o `find` precisa de dois argumentos: um ponto de partida e o que procurar. Por exemplo, para pesquisar todos os arquivos no diretório atual (e seus subdiretórios) cujo nome termina em `.jpg`, usariámos:

```
$ find . -name '*.jpg'  
./pixel_3a_seethrough_1.jpg  
./Mate3.jpg  
./Expert.jpg  
./Pentaro.jpg  
./Mate1.jpg  
./Mate2.jpg  
./Sala.jpg
```

```
./Hotbit.jpg
```

Essa busca corresponde a qualquer arquivo cujos últimos quatro caracteres do nome sejam `.jpg`, não importa o que venha antes, já que `*` é um curinga para “qualquer coisa”. No entanto, veja o que acontece se outro `*` for adicionado no *final* do padrão:

```
$ find . -name '*.*.jpg*'
./pixel_3a_seethrough_1.jpg
./Pentaro.jpg.zip
./Mate3.jpg
./Expert.jpg
./Pentaro.jpg
./Mate1.jpg
./Mate2.jpg
./Sala.jpg
./Hotbit.jpg
```

O arquivo `Pentaro.jpg.zip` (destacado acima) não foi incluído na lista anterior, porque embora contenha `.jpg` em seu nome, não corresponde ao padrão, já que há caracteres extras depois dele. O novo padrão significa “qualquer coisa `.jpg` qualquer coisa”, então ele passa a corresponder.



Lembre-se de que o parâmetro `-name` diferencia maiúsculas de minúsculas. Se quiser fazer uma pesquisa que não diferencia maiúsculas de minúsculas, use `-iname`.

A expressão `*.jpg` deve ser posta entre aspas simples, para evitar que o shell interprete o padrão como um comando. Experimente sem as aspas e veja o que acontece.

Por padrão, o `find` começa no ponto de partida e vai descendo pelos subdiretórios (e subdiretórios desses subdiretórios) encontrados. Para restringir esse comportamento, use os parâmetros `-maxdepth N`, onde `N` é o número máximo de níveis.

Para pesquisar apenas no diretório atual, usariamos `-maxdepth 1`. Suponha que você tem a seguinte estrutura de diretório:

```
directory
  clients.txt
  partners.txt -> clients.txt
  somedir
    anotherdir
    clients.txt
```

Para pesquisar dentro de `somedir`, você precisará usar `-maxdepth 2` (o diretório atual +1 nível abaixo). Para pesquisar dentro de `anotherdir`, `-maxdepth 3` seria necessário (o diretório atual +2 níveis abaixo). O parâmetro `-mindepth N` faz o contrário, pesquisando apenas em diretórios *no mínimo N* níveis abaixo.

O parâmetro `-mount` pode ser usado para evitar que o `find` mergulhe em sistemas de arquivos montados. Também é possível restringir a pesquisa a tipos específicos de sistemas de arquivos usando o parâmetro `-fstype`. Assim, `find /mnt -fstype exfat -iname "*report*`" buscaria somente dentro de sistemas de arquivos exFAT montados sob `/mnt`.

Buscando por atributos

Podemos usar os parâmetros abaixo para pesquisar arquivos com atributos específicos, como os que podem ser gravados por seu usuário, os que têm um conjunto específico de permissões ou que têm um determinado tamanho:

-user USERNAME

Encontra arquivos de que o usuário USERNAME é proprietário.

-group GROUPNAME

Encontra arquivos de que o grupo GROUPNAME é proprietário.

-readable

Encontra arquivos que podem ser lidos pelo usuário atual.

-writable

Encontra arquivos que podem ser gravados pelo usuário atual.

-executable

Encontra arquivos que podem ser executados pelo usuário atual. No caso dos diretórios, ele encontrará quaisquer diretórios em que o usuário pode entrar (permissão x).

-perm NNNN

Encontra quaisquer arquivos que tenham exatamente a permissão NNNN. Por exemplo, `-perm 0664` corresponde a quaisquer arquivos que o usuário e grupo podem ler e gravar e que outros podem ler (ou `rw-rw-r--`).

Adicione um `-` antes de NNNN para procurar por arquivos que tenham *pelo menos* a permissão especificada. Por exemplo, `-perm -644` encontraria arquivos que tenham permissões de ao menos 644 (`rw-r-r--`). Isso inclui arquivos com 664 (`rw-rw-r--`) ou mesmo 775 (`rwxrwx-r-x`).

-empty

Encontra arquivos e diretórios vazios.

-size N

Encontra quaisquer arquivos de tamanho N, onde N por padrão é um número de blocos de 512 bytes. Podemos adicionar sufixos a N para as outras unidades: Nc conta o tamanho em bytes, Nk em kibibytes (KiB, múltiplos de 1024 bytes), NM em mebibytes (MiB, múltiplos de 1024 * 1024) e NG para gibibytes (GiB, múltiplos de 1024 * 1024 * 1024).

Aqui também, podemos usar os prefixos + ou - (que aqui significam *maior que* e *menor que*) para buscar por tamanhos relativos. Por exemplo, `-size -10M` corresponderia a qualquer arquivo menor de 10 MiB.

Assim, para pesquisar arquivos em seu diretório inicial que contenham o padrão `report` (sem distinção entre maiúsculas e minúsculas) em qualquer parte do nome, tenham permissões 0644, foram acessados há 10 dias e cujo tamanho é de pelo menos 1 Mib, usariamos

```
$ find ~ -iname "*report*" -perm 0644 -atime 10 -size +1M
```

Busca por tempo

Além da busca por atributos, também é possível realizar buscas por data e hora, encontrando arquivos que foram acessados, tiveram seus atributos alterados ou foram modificados durante um determinado período de tempo. Os parâmetros são:

-amin N, -cmin N, -mmin N

Corresponde a arquivos que foram acessados, tiveram atributos alterados ou foram modificados (respectivamente) N minutos atrás.

-atime N, -ctime N, -mtime N

Corresponde a arquivos que foram acessados, tiveram atributos alterados ou foram modificados N*24 horas atrás.

Para `-cmin N` e `-ctime N`, qualquer alteração de atributo é levada em conta, incluindo mudanças nas permissões, leitura ou gravação no arquivo. Isso torna esses parâmetros especialmente poderosos, uma vez que praticamente qualquer operação envolvendo o arquivo fará com que ele corresponda aos parâmetros da busca.

O exemplo a seguir corresponderia a qualquer arquivo no diretório atual que foi modificado há menos de 24 horas e é maior que 100 MiB:

```
$ find . -mtime -1 -size +100M
```

Usando locate e updatedb

locate e updatedb são comandos que podem ser usados para encontrar rapidamente um arquivo correspondente a um padrão dado em um sistema Linux. Mas, ao contrário de find, locate não pesquisa por um padrão no sistema de arquivos: em vez disso, ele consulta um banco de dados construído com a execução do comando updatedb. Os resultados são muito velozes, mas podem ser imprecisos dependendo da data da última atualização do banco de dados.

A maneira mais simples de usar locate é simplesmente fornecer a ele um padrão a pesquisar. Por exemplo, para encontrar todas as imagens JPEG em seu sistema, você usaria locate jpg. A lista de resultados pode ser bastante longa, mas teria a seguinte aparência:

```
$ locate jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
/home/carol/Downloads/jpg_specs.doc
```

Quando buscamos pelo padrão jpg, o locate mostra qualquer coisa que contenha esse padrão, não importa o que venha antes ou depois. Você pode ver um exemplo disso no arquivo jpg_specs.doc na lista acima: ele contém o padrão, mas sua extensão não é jpg.

TIP Com locate, estamos procurando por padrões, e não por extensões de arquivo.

Em princípio, o padrão diferencia maiúsculas de minúsculas. Isso significa que arquivos contendo .JPG não seriam mostrados, pois o padrão está em letras minúsculas. Para evitar isso, passe o parâmetro -i para locate. Repetindo nosso exemplo anterior:

```
$ locate -i .jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate1.old.JPG
```

```
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
```

Observe que o arquivo **Mate1_old.JPG**, destacado em negrito acima, não estava presente na listagem anterior.

Para passar vários padrões para `locate`, basta separá-los com espaços. O exemplo abaixo faria uma busca sem distinção entre maiúsculas e minúsculas por quaisquer arquivos que correspondam aos padrões `zip` e `jpg`:

```
$ locate -i zip jpg
/home/carol/Downloads/Expert.jpg
/home/carol/Downloads/Hotbit.jpg
/home/carol/Downloads/Mate1.jpg
/home/carol/Downloads/Mate1_old.JPG
/home/carol/Downloads/Mate2.jpg
/home/carol/Downloads/Mate3.jpg
/home/carol/Downloads/OPENMSXPIHAT.zip
/home/carol/Downloads/Pentaro.jpg
/home/carol/Downloads/Sala.jpg
/home/carol/Downloads/gbs-control-master.zip
/home/carol/Downloads/lineage-16.0-20190711-MOD-quark.zip
/home/carol/Downloads/pixel_3a_seethrough_1.jpg
/home/carol/Downloads/jpg_specs.doc
```

Ao usar múltiplos padrões, você pode solicitar ao `locate` que exiba apenas os arquivos que correspondam a *todos* eles. Isso é feito com a opção `-A`. O exemplo a seguir mostraria qualquer arquivo que corresponda aos padrões `.jpg` e `.zip`:

```
$ locate -A .jpg .zip
/home/carol/Downloads/Pentaro.jpg.zip
```

Se quiser contar o número de arquivos que correspondem a um determinado padrão em vez de mostrar o caminho completo, você pode usar a opção `-c`. Por exemplo, para contar o número de arquivos `.jpg` em um sistema:

```
$ locate -c .jpg
```

1174

Um problema com o `locate` é que ele mostra apenas as entradas presentes no banco de dados gerado pelo `updatedb` (localizado em `/var/lib/mlocate.db`). Se o banco de dados estiver desatualizado, a saída poderá mostrar arquivos que foram excluídos desde a última vez em que foi atualizado. Uma maneira de evitar isso é adicionar o parâmetro `-e`, que fará com que ele verifique se o arquivo ainda existe antes de mostrá-lo na saída.

Obviamente, isso não fará com que os arquivos criados *após* a última atualização do banco de dados apareçam. Para isso será necessário atualizar o banco de dados com o comando `updatedb`. O tempo para a execução dessa operação dependerá da quantidade de arquivos em seu disco.

Controlando o comportamento de `updatedb`

O comportamento do `updatedb` pode ser controlado pelo arquivo `/etc/updatedb.conf`. Trata-se de um arquivo de texto no qual cada linha controla uma variável. As linhas em branco são ignoradas e as linhas que começam com o caractere `#` são tratadas como comentários.

PRUNEFS=

Quaisquer tipos de sistemas de arquivos indicados após este parâmetro não serão analisados pelo `updatedb`. A lista de tipos deve ser separada por espaços e os tipos em si não diferenciam maiúsculas de minúsculas, de forma que `NFS` e `nfs` são a mesma coisa.

PRUNENAMES=

Lista de nomes de diretórios separados por espaços que não deverá ser analisada pelo `updatedb`.

PRUNEWAYS=

Lista de nomes de caminhos que devem ser ignorados pelo `updatedb`. Os nomes de caminhos devem ser separados por espaços e especificados da mesma maneira como seriam mostrados pelo `updatedb` (por exemplo, `/var/spool /media`)

PRUNE_BIND_MOUNTS=

Esta é uma simples variável de `yes` ou `no`. Se definida como `yes`, as bind mounts (montagens de ligação: diretórios montados em outros locais com o comando `mount --bind`) serão ignoradas.

Encontrando binários, páginas de manual e código-fonte

O `which` é um comando muito útil que mostra o caminho completo para um executável. Por exemplo, se você quiser localizar o executável para `bash`, pode usar:

```
$ which bash
/usr/bin/bash
```

Se a opção `-a` for adicionada, o comando mostrará todos os nomes de caminho que correspondem ao executável. Observe a diferença:

```
$ which mkfs.ext3
/usr/sbin/mkfs.ext3
```

```
$ which -a mkfs.ext3
/usr/sbin/mkfs.ext3
/sbin/mkfs.ext3
```

TIP

Para descobrir quais diretórios estão no PATH, use o comando `echo $PATH`. Ele exibe (echo) o conteúdo da variável PATH (`$PATH`) no terminal.

`type` é um comando semelhante que mostra informações sobre um binário, incluindo sua localização e tipo. Basta usar `type` seguido do nome do comando:

```
$ type locate
locate is /usr/bin/locate
```

O parâmetro `-a` funciona da mesma maneira que em `which`, mostrando todos os nomes de caminho que correspondem ao executável. Assim:

```
$ type -a locate
locate is /usr/bin/locate
locate is /bin/locate
```

E o parâmetro `-t` mostra o tipo de arquivo do comando que pode ser `alias`, `keyword`, `function`, `builtin` ou `file`. Por exemplo:

```
$ type -t locate
file

$ type -t ll
alias
```

```
$ type -t type  
type is a built-in shell command
```

O comando `whereis` é mais versátil e, além dos binários, também pode ser usado para mostrar a localização das páginas do manual ou mesmo o código-fonte de um programa (se disponível em seu sistema). Basta digitar `whereis` seguido do nome binário:

```
$ whereis locate  
locate: /usr/bin/locate /usr/share/man/man1/locate.1.gz
```

Os resultados acima incluem binários (`/usr/bin/locate`) e páginas de manual comprimidas (`/usr/share/man/man1/locate.1.gz`).

Podemos filtrar rapidamente os resultados usando opções de linha de comando como `-b`, para limitá-los apenas aos binários, `-m`, para limitá-los apenas a páginas de manual, ou `-s`, para limitá-los apenas ao código-fonte. Repetindo o exemplo acima, os resultados seriam:

```
$ whereis -b locate  
locate: /usr/bin/locate  
  
$ whereis -m locate  
locate: /usr/share/man/man1/locate.1.gz
```

Exercícios Guiados

- Imagine que um programa precisa criar um arquivo temporário de uso único, que não será novamente necessário após o encerramento do programa. Qual seria o diretório correto para criar esse arquivo?

- Qual é o diretório temporário que *deve* ser limpo durante o processo de inicialização?

- Usando `find`, pesquise apenas no diretório atual por arquivos que são graváveis pelo usuário, foram modificados nos últimos 10 dias e são maiores que 4 GB.

- Usando `locate`, encontre todos os arquivos contendo os padrões `report` e `updated`, `update` ou `updating` em seus nomes.

- Como descobrir onde a página de manual para `ifconfig` está armazenada?

- Qual variável precisa ser adicionada a `/etc/updatedb.conf` para que `updatedb` ignore sistemas de arquivos `ntfs`?

- Um administrador de sistema deseja montar um disco interno (`/dev/sdc1`). De acordo com o FHS, em qual diretório este disco deve ser montado?

Exercícios Exploratórios

1. Quando `locate` é usado, os resultados são puxados de um banco de dados gerado por `updatedb`. No entanto, este banco de dados pode estar desatualizado, fazendo com que o `locate` mostre arquivos que não existem mais. Como fazer com que `locate` mostre apenas arquivos existentes em sua saída?

[Resposta]

2. Encontre qualquer arquivo no diretório ou subdiretórios atuais até 2 níveis abaixo, excluindo sistemas de arquivos montados, que contenham o padrão `Status` ou `statute` em seus nomes.

[Resposta]

3. Limitando a pesquisa aos sistemas de arquivos `ext4`, encontre quaisquer arquivos sob `/mnt` que tenham no mínimo permissões de execução para o grupo, sejam legíveis para o usuário atual e tenham tido qualquer atributo alterado nas últimas 2 horas.

[Resposta]

4. Encontre arquivos vazios que foram criados há mais de 30 dias e estão pelo menos dois níveis abaixo do diretório atual.

[Resposta]

5. Considere que os usuários `carol` e `john` fazem parte do grupo `mkt`. Encontre no diretório pessoal de `john` quaisquer arquivos que também possam ser lidos por `carol`.

[Resposta]

Resumo

Nesta lição, você aprendeu sobre a organização básica do sistema de arquivos em uma máquina Linux, de acordo com o FHS, e como encontrar binários e arquivos, seja por nome ou por atributos. Os seguintes comandos foram discutidos nesta lição:

find

Um comando versátil usado para encontrar arquivos e pastas segundo uma variedade de critérios de pesquisa.

locate

Um utilitário que usa um banco de dados local contendo a localização dos arquivos armazenados localmente.

updatedb

Atualiza o banco de dados local usado pelo comando `locate`.

which

Exibe o caminho completo até um executável.

whereis

Exibe a localização das páginas de manual, binários e código-fonte no sistema.

type

Exibe a localização de um binário e de que tipo de aplicativo se trata (um programa instalado, um programa interno do Bash e assim por diante).

Respostas aos Exercícios Guiados

- Imagine que um programa precisa criar um arquivo temporário de uso único, que não será novamente necessário após o encerramento do programa. Qual seria o diretório correto para criar esse arquivo?

Como não precisamos mais do arquivo depois que o programa for encerrado, o diretório correto é /tmp.

- Qual é o diretório temporário que *deve* ser limpo durante o processo de inicialização?

O diretório é /run ou, em certos sistemas, /var/run.

- Usando o `find`, pesquise apenas no diretório atual por arquivos que são graváveis pelo usuário, foram modificados nos últimos 10 dias e são maiores que 4 GB.

Para isso, precisamos dos parâmetros `-writable`, `-mtime` e `-size`:

```
find . -writable -mtime -10 -size 4G
```

- Usando `locate`, encontre todos os arquivos contendo os padrões `report` e `updated`, `update` ou `updating` em seus nomes.

Como `locate` precisa encontrar todos os padrões correspondentes, use a opção `-A`:

```
locate -A "report" "updat"
```

- Como descobrir onde a página de manual para `ifconfig` está armazenada?

Use o parâmetro `-m` para o `whereis`:

```
whereis -m ifconfig
```

- Qual variável precisa ser adicionada a `/etc/updatedb.conf` para que `updatedb` ignore sistemas de arquivos `ntfs`?

A variável é `PRUNEFS=` seguida pelo tipo do sistema de arquivos: `PRUNEFS=ntfs`

- Um administrador de sistema deseja montar um disco interno (`/dev/sdc1`). De acordo com o

FHS, em qual diretório este disco deve ser montado?

Na prática, o disco pode ser montado em qualquer lugar. Porém, o FHS recomenda que montagens temporárias sejam feitas em /mnt

Respostas aos Exercícios Exploratórios

1. Quando `locate` é usado, os resultados são puxados de um banco de dados gerado por `updatedb`. No entanto, este banco de dados pode estar desatualizado, fazendo com que o `locate` mostre arquivos que não existem mais. Como fazer com que `locate` mostre apenas arquivos existentes em sua saída?

Adicione o parâmetro `-e` a `locate`, como em `locate -e PATTERN`.

2. Encontre qualquer arquivo no diretório ou subdiretórios atuais até 2 níveis abaixo, excluindo sistemas de arquivos montados, que contenham o padrão `Status` ou `statute` em seus nomes.

Lembre-se de que para `-maxdepth` também devemos levar em conta o diretório atual, de forma que queremos três níveis (o atual, mais 2 níveis para baixo):

```
find . -maxdepth 3 -mount -iname "*statu*"
```

3. Limitando a pesquisa aos sistemas de arquivos `ext4`, encontre quaisquer arquivos sob `/mnt` que tenham no mínimo permissões de execução para o grupo, sejam legíveis para o usuário atual e tenham tido qualquer atributo alterado nas últimas 2 horas.

Use o parâmetro `-fstype` de `mount` para limitar a pesquisa a tipos específicos de sistemas de arquivos. Um arquivo legível pelo usuário atual teria ao menos `4` no primeiro dígito de permissões, e um executável pelo grupo teria ao menos `1` no segundo dígito. Como não estamos preocupados com as permissões de outros, podemos usar `0` no terceiro dígito. Use `-cmin N` para filtrar alterações recentes de atributos, lembrando que `N` é especificado em minutos. Assim:

```
find /mnt -fstype ext4 -perm -410 -cmin -120
```

4. Encontre arquivos vazios que foram criados há mais de 30 dias e estão pelo menos dois níveis abaixo do diretório atual.

O parâmetro `-mindepth N` pode ser usado para limitar a pesquisa a pelo menos `N` níveis abaixo, mas é necessário incluir o diretório atual na contagem. Use `-empty` para procurar por arquivos vazios e `-mtime N` para ver a data e hora de modificação. Assim:

```
find . -empty -mtime +30 -mindepth 3
```

5. Considere que os usuários `carol` e `john` fazem parte do grupo `mkt`. Encontre no diretório pessoal

de john quaisquer arquivos que também possam ser lidos por carol.

Considerando que os dois são membros do mesmo grupo, precisamos de pelo menos um r (4) nas permissões de grupo e não estamos preocupados com os outros. Assim:

```
find /home/john -perm -040
```

Imprint

© 2022 by Linux Professional Institute: Materiais Didáticos, “LPIC-1 (101) (Version 5.0)”.

PDF gerado: 2022-06-03

Este trabalho está licenciado sob Creative Commons Licença Atribuição-NãoComercial-SemDerivações 4.0 Internacional (CC BY-NC-ND 4.0). Para ver uma cópia desta licença, visite

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Embora o Linux Professional Institute tenha agido de boa fé para garantir que as informações e instruções contidas neste trabalho sejam exatas, o Linux Professional Institute isenta-se de qualquer responsabilidade por erros ou omissões, incluindo, sem limitações, a responsabilidade por danos resultantes do uso ou confiança nesta obra. O uso das informações e instruções contidas neste trabalho deve ser feito por sua própria conta e risco. Se as amostras de código ou outras tecnologias contidas ou descritas neste trabalho estiverem sujeitas a licenças de código aberto ou direitos de propriedade intelectual de terceiros, é sua responsabilidade garantir que seu uso esteja em conformidade com tais licenças e/ou direitos.

Os Materiais Didáticos da LPI são uma iniciativa do Linux Professional Institute (<https://lpi.org>). Os Materiais Didáticos e suas traduções estão disponíveis em <https://learning.lpi.org>.

Para perguntas e comentários sobre esta edição, bem como sobre todo o projeto, escreva para: learning@lpi.org.